Estudi GPGPU dedicat al processament de neuroimatges

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per César Allande Álvarez
i dirigit per Diego Javier Mostaccio Mancini

Bellaterra, 19 de Juny de 2009.

Universitat Autònoma de Barcelona

etse
Escola Tècnica Superior d'Enginyeria

El sotasignat, Diego Javier Mostaccio Mancini

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en César Allande Álvarez

I per tal que consti firma la present.

Signat: Diego Javier Mostaccio Mancini

Bellaterra, 19 de Juny de 2009

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, Yolanda Vives Gilabert

de l'empresa, PIC (Port d'Informació Científica)

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat en l'empresa sota la seva supervisió mitjançant conveni entre PIC (Port d'Informació Científica) i DACSO (Departament d'arquitectura de Computadors i Sistemes Operatius) de la Universitat Autònoma de Barcelona.

Així mateix, l'empresa en té coneixement i dóna el vist-i-plau al contingut que es detalla en aquesta memòria.

Signat: Yolanda Vives Gilabert

Cerdanyola, 19 de Juny de 2009

Taula de Continguts

1 Introducció

El PIC(Port d'Informació Científica) és un centre innovador que dóna suport científic a grups que requereixen recursos de computació i d'emmagatzemament de grans quantitats de dades per mitjà de les tecnologies emergents conegudes com Grid. El PIC té responsabilitats en el projecte europeu EGEE(Enabling Grids for E-sciencE) i opera el centre espanyol Tier-1 per a processament de dades del LHC (Large Hadron Collider). També participa en projectes d'altres àmbits científics com l'investigació mèdica, grup dintre del que s'inscriu aquest projecte.

L'objectiu d'aquest projecte és l'estudi i implementació d'una plataforma de càlcul GPGPU (General-Purpose computing on Graphics Processing Units), una nova tecnologia que aprofita el rendiment de les targetes gràfiques d'última generació, que es vol aplicar al càlcul científic com classificadors biomèdics o post-processament de neuroimatges amb Matlab, on es pretén treure profit de l'acceleració GPGPU amb la tecnologia subministrada per NVIDIA, la plataforma anomenada CUDA (Compute Unified Device Architecture) amb que identificar els processos amb alt cost computacional i adaptar-los per tal de fer càlculs de speedup respecte les versions actuals que treballen directament amb la CPU.

2 Estat de l'art

Aquest projecte està definit dins l'àmbit de les GPUs i la Biomedicina. S'explica a continuació quin és el nivell de desenvolupament actual en aquests dos camps.

2.1 El món de les GPUs

Per norma general les unitats que realitzen el càlcul numèric són les CPUs, ja sigui en computació serie o paral·lela. En la actualitat les necessitats de càlcul s'orienten a estratègies de processament de dades en paral·lel i per això s'estableixen estratègies en infraestructures de Griding o Clústering per aprofitar la capacitat de còmput de moltes CPUs treballant conjuntament. Sabem que tecnològicament les CPUs dupliquen cada 18 mesos la seva capacitat de processament seguint un creixement aproximat a la llei de Moore. Un gran limitador de la capacitat de càlcul de les CPUs a l'hora de realitzar operacions en punt flotant és la utilització de gran part dels transistors integrats en el xip per la realització d'optimitzacions de codi seqüencial, e.g. Loop unrolling, Branch Prediction.

A diferència les GPU no estan orientades al control de flux i no treballen com un processador d'àmbit general, la seva necessitat principal és el càlcul numèric en punt flotant amb la màxima taxa de processament possible. Això resulta molt interessant a l'hora de fer servir la GPU com a coprocessador matemàtic.

Si a començaments de 2003 les CPUs i les GPUs ja oferien un rendiment en punt flotant molt similar, a finals de 2006 la família G80 de NVIDIA excedia 60 vegades la capacitat del màxim rendiment teòric del Core 2 Duo de Intel. Des de que la capacitat de processament de les GPUs va sobrepassar la de les CPUs s'han aplicat diverses estratègies per treure'n profit d'aquest recurs. Existeixen actualment diferents llenguatges i llibreries que permeten fer servir el processador gràfic com a coprocessador matemàtic, per la naturalesa implícita d'aquestes targetes fa que al estar orientades a treballar amb gràfics la seva programació amb propòsit general s'ha de fer amb instruccions i estructures de dades específiques del món dels gràfics per computador, ja que s'està utilitzant un processador amb una finalitat completament diferent a la

que havia estat dissenyada, i això provoca que la seva programació sigui més complicada del que ho és amb una CPU.

Les empreses fabricants de GPUs en veure que poden explotar l'àrea de mercat de la supercomputació aposten per tecnologies aplicades al GPGPU. La empresa AMD/ATI proposa la tecnologia ATI Stream, anteriorment ho va fer amb CTM (Close to Metal) i per altra banda NVIDIA suporta el llenguatge CUDA a partir de la serie G80, G80M (utilitzada en portàtils), Tesla i Quadro. CUDA resulta molt interessant per a un programador ja que si altres models treballen a baix nivell, aquest es programa com una extensió de ANSIC, i la companyia NVIDIA assegura que la corba d'aprenentatge per l'usuari és molt pronunciada (molt aprenentatge en poc temps). Un altre dels avantatges de CUDA és que proporciona una extensió pel software Matlab (Matrix Laboratory) que és un software matemàtic molt utilitzat en el món científic i concretament en les aplicacions del grup de Neuroimatge del PIC (Port d'Informació Científica), que és el centre de treball on es desenvolupa aquest projecte.

2.2 GPU aplicada a la Biomedicina i la Neuroimatge

Existeixen diverses aplicacions de GPGPU aplicades al món de la Biomedicina i la Neuroimatge on es posa de manifest el major rendiment aconseguit respecte als processos existents amb CPU, per això s'han fet servir diferents components hardware i tecnologies GPGPU. És important destacar que la major part del temps d'estudi radica en trobar la millor forma de paral·lelitzar els processos amb més intensitat aritmètica perquè el pes del còmput recaigui a la GPU.

Un primer exemple de GPU aplicada al món de la Biomedicina és l'estudi [*Antonio Ruiz et al.*] on s'investiga la capacitat de processament entre CPU i GPU per l'anàlisi de fenotips a través del color. Per les proves amb CPU, els autors fan servir un entorn amb Visual Studio .NET sota Windows XP, i per programar la GPU es fa servir el llenguatge Cg (C for graphics de NVIDIA) i OpenGL 2.0 amb algunes extensions. En aquest document es fa una comparativa per diferents famílies de CPUs i GPUs existents entre els anys 2004 i 2006. La comparativa conclou que la

millora de les GPUs vers les CPUs de la mateixa generació arriba a un guany entre 4x i 100x depenent de la naturalesa dels càlculs, la família del hardware i els recursos de programació emprats.

Un segon exemple és *[Anthony Gregerson]* on es realitza una implementació d'un fast Magnetic Ressonance Imaging que aplicant diferents tècniques, aconsegueix un speedup de 114x respecte a la implementació amb CPU (Geforce 8800 GTX vs. Core 2 Duo). El procés que es segueix en aquest estudi consisteix en l'estudi de l'algorisme FFT (Fast Fourier Transform) per a un entorn GPGPU amb CUDA, el posterior desenvolupament del software i finalment es realitzen proves dels diversos algorismes amb les que s'han realitzat les comparatives.

Com hem vist, les noves targetes gràfiques aconsegueixen més altes prestacions en el càlcul en punt flotant, amb l'afegit que tenen relativament un baix cost, això possibilita implantar centres de càlcul, inclús portables (amb ordinadors portàtils), físicament dins d'hospitals universitaris, centres d'investigació biomèdica, etc. Les tècniques de clústering o griding també se'n poden veure beneficiades del coprocessament GPU, com per exemple la proposta NVIDIA amb la solució especialitzada per servidors anomenada Tesla, o la distribució pel sistema operatiu Rocks Cluster (versió 4.3) de clústering basat en Linux consistent en un roll CUDA, que serveix per a la distribució del sotware als nodes del clúster.

Actualment es fa servir al departament de Neuro del centre PIC el paquet SPM per examinar diferències del cervell amb tecnologies de captació de neuroimatges com MRI (Magnetic Resonance Imaging) o PET (Positron Emission Tomography). Per realitzar els càlculs es fan servir funcions executades en CPU, ja que fins a la data no existeix una versió SPM amb aprofitament de la GPU. El fet que el paquet SPM estigui en part codificat en llenguatge M (Matlab) i en part amb algunes rutines precompilades proporciona la possibilitat d'investigar quins processos es poden adaptar amb CUDA.

Un exemple d'un procediment existent que pot treure benefici del coprocessament GPU és el d'un proces biomèdic, desenvolupat al PIC, basat en un tractament Voxel-Based Morphometry (VBM) que serveix per identificar diferències estructurals en el cervell, per grups amb enfermetats neurodegeneratives, i que fa servir els recursos del paquet SPM. Actualment, un estudi mitjà de VBM, te un temps de còmput que pot trigar entre 15 o 20 hores.

Al PIC no només es treballa amb el paquet SPM, s'està desenvolupant un classificador amb SVM (Support Vector Machine) per la identificació de marcadors neurodegeneratius, que també té un cost de còmput elevat i que també pot treure benefici amb l'acceleració via GPGPU.

Al centre PIC es tenen molt processos Biomèdics i de Neuroimatge que poden obtenir una optimització en el temps d'execució si s'aplica un ús de coprocessament matemàtic amb GPU, i aplicant bones estratègies de paral·lelització que reportin beneficis en aquest àmbit del món Científic.

3 Estudi de Viabilitat

Com s'ha explicat, les tecnologies GPGPU han demostrat que és una estratègia viable i que optimitzen processos científics, com en *[Anthony Gregerson], [Friedemann Rößler et al.],[Antonio Ruiz et al.]* i d'altres. També aporta una solució bastant econòmica, i per això s'ha escollit per optimitzar els processos biomèdics i de neuroimatge que es fan servir al PIC. El motiu d'escollir CUDA entre totes les tecnologies GPGPU és perquè en comparació amb d'altres tecnologies aquesta treballa a més alt nivell, això implica un menor temps d'aprenentatge i a més existeixen bastants exemples i documentació. Això ens permet planificar major part del temps a l'estudi de la paral·lelització dels algorismes. També, a la data d'elaboració del projecte, és la tecnologia més suportada al mercat i amb més projecció, ja que l'empresa desenvolupadora actualment és líder en el mercat dels xips gràfics

En el projecte, la viabilitat ve determinada per l'obtenció de resultats que ens permetin avaluar els beneficis d'aplicar GPGPU respecte a les versions actuals i desenvolupar un prototip que permeti millorar el temps de càlcul d'un procés que es faci servir actualment al PIC.

3.1 Entorn de desenvolupament

Per desenvolupar el projecte es posen a disposició del desenvolupador uns recursos hardware i software. Detallem a continuació les especificacions tècniques:

3.1.1 Recursos Hardware

Al llarg del projecte s'han fet servir diferents estacions de treball, tant per desenvolupar com per fer proves en els temps d'execució. En les proves desenvolupades s'especifica amb quines d'aquestes estacions s'han fet les comparatives.

Ordinadors

1. HW:01 - Dell XPS 730x, estació de treball principal.
    - Inte Core 2 Extreme CPU x9650@ 3.00GHz (suport de 64 bits i 4 cores)
    - Dues Targetes gràfica NVIDIA GeForce 9800 GT, PCIe amb suport CUDA
    - Memòria DDR2 de 4 Gigabytes.

2. HW:02 - Dell Laptop Precision M6300, estació portable per realitzar proves de temps comparatives amb CUDA
    - Intel Core 2 Duo T7700 @ 2,40GHz
    - NVIDIA Quadro FX 1600M, 512 Mbytes
    - Memòria RAM de 2 Gigabytes

3. HW:03 - Thosiba Laptop A100-999, estació portable, sense suport CUDA, per realitzar proves de temps de càlcul del paquet SPM respecte l'estació de treball principal.
    - Intel Core 2 Duo T5500 @ 1,6GHz (suport de 64 bits i 2 cores)
    - Targeta gràfica ATI
    - Memòria DDR2 de 2 Gigabytes.

4. HW:04 - DELL Laptop Latitude D430, estació portable sense suport CUDA.
    - Intel Core 2 Duo CPU U7700 @ 1,33GHz
    - Targeta gràfica Intel GMA 950, 224MBytes
    - Memòria RAM de 2GBytes

5. HW:05 – Estació de treball per peces
    - Intel Pentium 4 @ 3,4GHz
    - Targeta gràfica NVIDIA GeForce 9800 GT, PCIe amb suport CUDA
    - Memòria RAM de 2GBytes

Descripció targetes gràfiques

- GeForce 9800GT

Processadors de tipus “stream”	112
Rellotge central (MHz)	600 MHz
Rellotge unitats shader (MHz)	1500 MHz
Rellotge de memòria	900 MHz
Quantitat de memòria	512MB
Interfície de memòria	256-bit GDDR3
Ample de banda de memòria (GB/s)	57,6

Taula 1: Característiques del xip gràfic GeForce 9800GT

- Quadro FX 1600M

Processadors de tipus “stream”	32
Rellotge central (MHz)	625 MHz
Rellotge unitats shader (MHz)	1250 MHz
Rellotge de memòria	800 MHz
Quantitat de memòria	512MB
Interfície de memòria	128-bit GDDR3
Ample de banda de memòria (GB/s)	25,6

Taula 2: Característiques del xip gràfic Quadro FX 1600M

3.1.2 Recursos Software

S'ha fet servir un sistema operatiu Linux, ja que es va comprovar inicialment que aquest està plenament suportat per la resta de tecnologies emprades.

- Sistema operatiu Linux Fedora 10
- gcc, gcc-c++, compiladors de llenguatge c i c++
- NVIDIA Drivers per linux amb suport CUDA
- CUDA Toolkit, amb el compilador nvcc
- CUDA SDK v2.0 for Linux, amb exemples CUDA
- CUDA plugin for Matlab, llibreria per executar CUDA en un entorn Matlab
- CUDA Visual profiler, eina per extreure informació sobre l'execució de CUDA
- Matlab, amb llicència
    - Image Processing Toolbox, developed by Mathworks. Per tractament d'imatges
    - MRI_TOOLBOX developed by CNSP in Flinders University, Australia. Permet llegir imatges de tipus NIfTI
- SPM5, Statistical Parametring Mapping
- Imatges en format DICOM. Amb ressonàncies magnètiques de pacients, degudament anonimitzades, per desenvolupar l'estudi VBM.
- Openoffice 3.1, per redactar documents de la memòria i realitzar les presentacions.
- LaTeX, per la redacció dels document de definició inicial del projecte i de viabilitat.
- Planner 0.14.3, per elaborar el diagrama Gantt amb la planificació temporal

3.2 Planificació temporal

El projecte està definit per una realització en 620 hores. Aquestes es realitzaran al centre de treball del PIC (Punt d'Informació Científica), amb una assistència planificada de 4 hores per dia. L'inici del projecte s'ha realitzat el 3 de Novembre de 2008, i la data final està prevista pel 31 de Juny de 2009. La planificació s'ha dividit en diverses fites, a l'acabament de cadascuna, es procedirà a fer un informe i presentació dels resultats davant el departament de Neuro del PIC. Els contingut dels informes dependran segons la seva naturalesa. En la *imatge 1* tenim la planificació inicial del projecte.

3.2.1 Planificació inicial

Es mostren a continuació les tasques i deliverables. En el diagrama de Gantt es mostren les fites al centre PIC, i en el segon bloc les fites de control amb el tutor de la UAB.

Fita 0 - Instal·lar l'entorn de treball. Manual d'instal·lació del software necessari. *Mes 1 (Novembre). Annexe I*

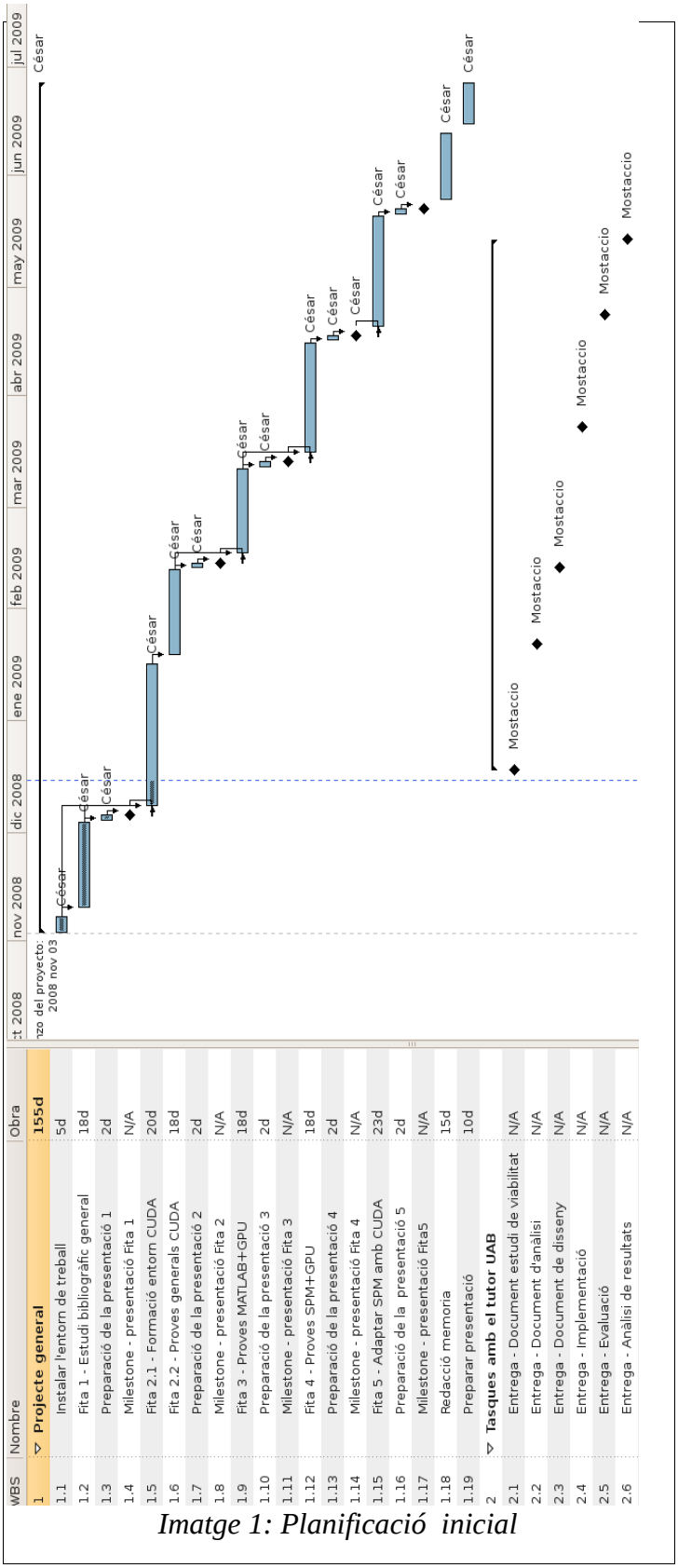
Fita 1 - Estudi bibliogràfic general: Exposició en detall de l'estat de l'art i de les vies de futur. *Mes 1 (Novembre)*

Fita 2 - Formació i proves en CUDA: Introducció a la programació en CUDA. Estudi de casos i anàlisi de resultats. *Mes 2-3 (Desembre-Gener)*

Fita 3 - Proves específiques MATLAB amb GPU: Integració de CUDA amb MATLAB i proves generals. Estudi de casos i anàlisi de resultats. *Mes 4 (Febrer)*

Fita 4 - Proves específiques SPM5 amb GPU: Adaptació de mòduls de SPM en l'entorn MATLAB per aprofitament amb CUDA. Estudi de casos i anàlisi de resultats. Estudi dels algorismes utilitzats a SPM i determinar l'estratègia per paral·lelitzar el proces. *Mes 5 (Març)*

Fita 5 - Adaptar SPM5 amb CUDA: Adaptar un paquet de SPM5 perquè utilitzi la plataforma CUDA, i poder fer comparatives entre execució normal i execució CUDA amb diferents models de GPUs. *Mes 6 (Abril)*



Imatge 1: Planificació inicial

3.3 Costos

La *taula 3* representa els costos del projecte, dividit en directes (recursos hardware, software i humans) i indirectes (derivats de infraestructura, despeses de llum, telèfon, coordinació, gestió, ...)

COSTOS DIRECTES		**20.610,15 €**
Recursos Hardware		***7.846,75 €***
HW:01	Dell XPS 730x	2.911,75 €
	+ (x2)Targeta gràfica NVIDIA 9800GT	300,00 €
HW:02	Dell Laptop Precision M6300	2.972,00 €
HW:03	Thosiba Laptop A100-999	600,00 €
HW:04	DELL Laptop Latitude D430	513,00 €
HW:05	Pentium IV	400,00 €
	+ Targeta gràfica NVIDIA 9800GT	150,00 €
Recursos Software		***2.950,00 €***
Matlab	Matlab license	1.950,00 €
	Image Processing Toolbox	1.000,00 €
Reursos Humans		**9.813,40 €**
Treballador a temps parcial (4hores)	8 mesos per 560€/mes	4.480,00 €
Supervisor	20% del temps (40.000€ anuals) durant 8 mesos	5.333,40 €
COSTOS INDIRECTES		**2.915,35 €**
Representen un 20% dels costos directes		**2.915,35 €**
COST TOTAL DEL PROJECTE (DIRECTES+INDIRECTES)		**23.525,50 €**

Taula 3: Estimació del cost total de projecte

4 Anàlisi i disseny

Aquest apartat reflexa primerament l'estudi del software SPM5 (Statistical Parametric Mapping versió 5) i a continuació l'anàlisi del mòdul SPM que s'ha decidit implementar a causa de les conclusions extretes en el primer anàlisi. El mòdul d'estudi correspon al procés de segmentació. Ha estat necessari fer aquest anàlisi previ, per començar a desenvolupar les optimitzacions amb CUDA, ja que s'ha necessitat detectar els punts crítics del codi SPM5 durant l'execució d'un procés típic de neuroimatge, concretament el tractament VBM(Voxel Based Morphometry), per seguidament, poder desenvolupar el prototip d'execució paral·lela d'un dels mòduls amb GPU. Els temps obtinguts durant aquest apartat reflexen les dades que justifiquen la decisió de realitzar el prototip sobre el procés de segmentació.

4.1 Anàlisi del tractament de VBM

El tractament VBM serveix per detectar diferències estructurals en el cervell, i aquest és un procés elaborat amb SPM.

El producte SPM és software lliure, distribuït sota els termes de la GNU GPL (GNU General Public License), de la versió 2 o qualsevol superior. Això ens permet accedir al codi font per realitzar les nostres pròpies modificacions, un dels requisits de la GNU GPL és mantenir la referència als autors del codi original i a la mateixa llicència, en cas de que vulguem redistribuir el nou codi a una tercera part. El procediment que s'ha seguit per fer aquest endavant:

- S'ha reproduït un procediment VBM amb l'entorn de finestres.
- S'han investigat les funcions que s'executen en cada procés de l'entorn gràfic. Ens hem centrat en cada un dels processos, utilitzats en un model concret de VBM, per fer un endavant en profunditat dels temps d'execució i definir el trossos de codi més susceptibles de ser paral·lelitzats.

Per iniciar aquest estudi, s'ha reproduït el tutorial de règim intern del centre PIC per tal de familiaritzar-se amb l'entorn SPM, que és el software que suporta el model VBM. Aquest model consisteix en el desenvolupament d'un model estadístic a partir de dos conjunts d'imatges. Un dels

processos típics és el modelatge d'un patró que permet als metges el diagnòstic de l'evolució de la malaltia d'un pacient. Per desenvolupar aquest patró es necessiten dos grups d'imatges, el primer grup són imatges de pacient que pateixen la malaltia en qüestió, i el segon grup de subjectes que no presenten la patologia. La finalitat del procés és obtenir un model morfològic, basat en estadística, de les regions del cervell que es veuen afectades per la malaltia. La degradació del teixit cerebral es presenta en malalties neurodegeneratives com per exemple l'Alzheimer o Parkinson.

4.1.1 Execució del model VBM

Les imatges de treball es generen en hospitals en la *taula 4* tenim maquinari especialitzat en l'adquisició *imatges 2 i imatge 3*, i els arxius que es proporcionen estan en format DICOM (Digital Imaging and Communication in Medicine). Aquest és el format estàndard reconegut internacionalment per l'intercanvi d'imatges mèdiques. L'estàndard DICOM especifica cóm tractar, emmagatzemar, imprimir i transmetre imatges mèdiques, inclou una definició de format de fitxer i protocols de comunicació. Els drets del format pertanyen a National Electrical Manufacturers Association (NEMA), i va ser desenvolupat pel DICOM Standards Committee format per membres de la NEMA.

Imatge 2: MRI Siemens Avanto 1,5 Tesla

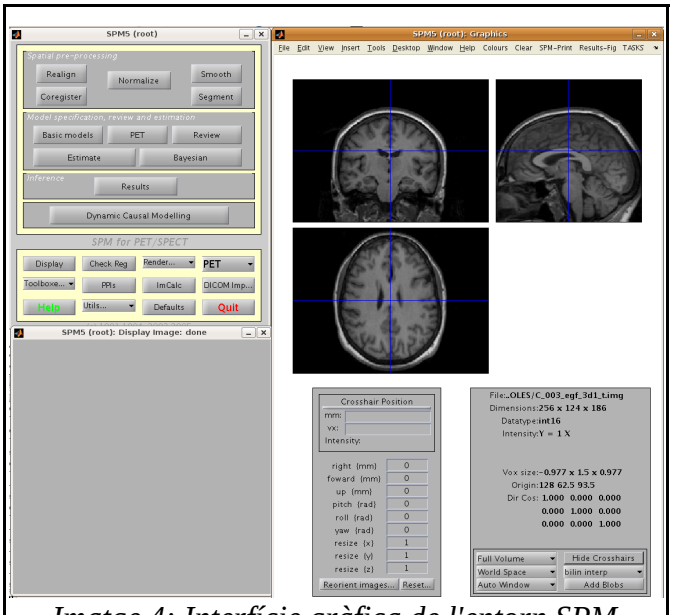
Imatge 3: Estació de treball mèdica Keosys

Taula 4: Adquisició d'imatges

Per poder començar l'execució del model, ens han proporcionat imatges en format DICOM dividides en dos grups. Un grup de pacients amb Alzheimer i l'altre de subjectes que no presenten la malaltia, anomenat grup de control. A continuació s'ha efectuat el procés de VBM en la pròpia màquina amb que hem obtingut un model estadístic basat en VBM. Cal dir que els resultats del model generat no tenen validesa i han estat descartats, ja que habitualment aquest procés es realitza amb 2 conjunts molts grans d'imatges, i nosaltres només hem agafat quatre imatges per grup.

4.1.2 Descripció del procés

L'entorn SPM proporciona una interfície visual que permet llançar els processos (jobs), podem veure a la *imatge 4* l'interfície gràfica d'SPM. Per executar SPM només s'ha d'incloure al PATH de Matlab la carpeta amb les llibreries SPM. Per afegir el PATH l'usuari pot accedir mitjançat File --Set Path, i en la finestra Add With Subfolder, es seleccionarà la carpeta contenidora d'SPM.

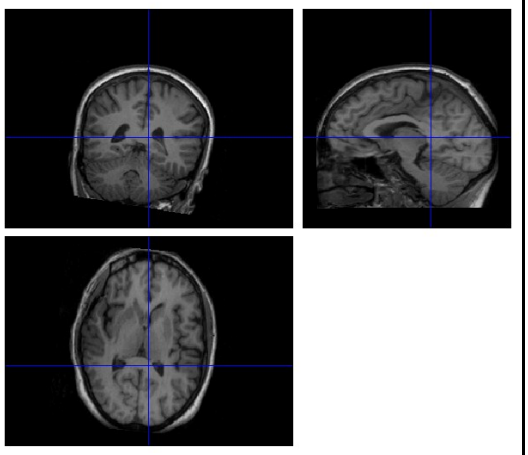


Imatge 4: Interfície gràfica de l'entorn SPM (Statistical Parametric Mapping)

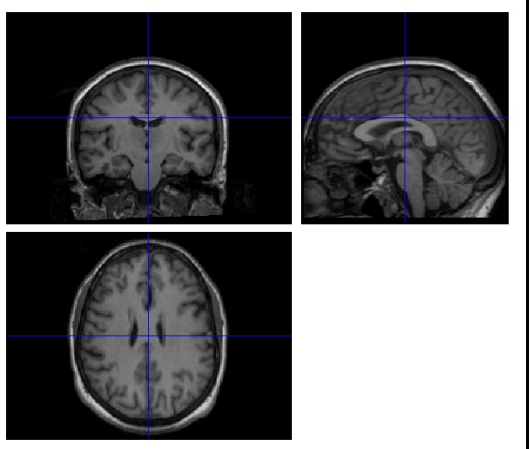
El tutorial especifica els passos a seguir, explicats de forma molt detallada, ens indica també els paràmetres de configuració de cada una de les fases. Els passos es poden fer dins l'entorn SPM i es descriuen a continuació:

Conversió de format de l'estàndard DICOM al format nadiu Analyze o NIfTI. El format Analyze és el format de la versió SPM2 i el formen dos arxius .hdr i .img. Per la versió SPM5 es fa servir el format NIfTI amb extensió .nii.

1. Classificació en dos grups d'estudi: Grup de control i grup d'estudi.
2. Fase de reorientació i normalització. El paquet SPM5 proporciona eines per rotar, escalar i centrar les imatges en 3D. Es fa de forma manual per ubicar dins el mateix espai estereotàxic per totes les imatges ajustant l'eix en un punt concret del cervell, que en el nostre cas serà la comissura anterior. Com es pot veure en la *taula 5*, en *imatge 5 i imatge 6*. La normalització ens ajusta, a un patró establert, totes les imatges perquè tinguin la mateixa dimensionalitat i es transformen per adaptar-se al model. Això es perquè el gruix i forma des cervells no és la mateixa i cal fer-ho per comparar la densitat de les escorces cerebrals.
3. Segmentació de les imatges. Seleccionant el conjunt de imatges i unes probabilitats a priori, s'especifiquen els paràmetres de segmentació. Aquesta fase separa els teixits d'estudi, i es generen tres arxius, un amb la matèria gris, un amb la matèria blanca (més greixosa i en la imatge és veu més brillant) i el líquid cefaloraquidi.
4. Smooth, és un filtre que permet suavitzar les imatges. Aquest filtre fa un promig de la intensitat d'un vòxel amb els vòxels adjacents.
5. Basic models, ens genera el model estadístic.
6. Results, amb l'estimació calculada anteriorment, aquest procés mostra els resultats del model generat. El resultat és un model tridimensional que representa les zones del cervell que han patit una degradació.



Imatge 5: Imatge original obtinguda de la Ressonància Magnètica



Imatge 6: Imatge correctament alineada i centrada en la comissura anterior

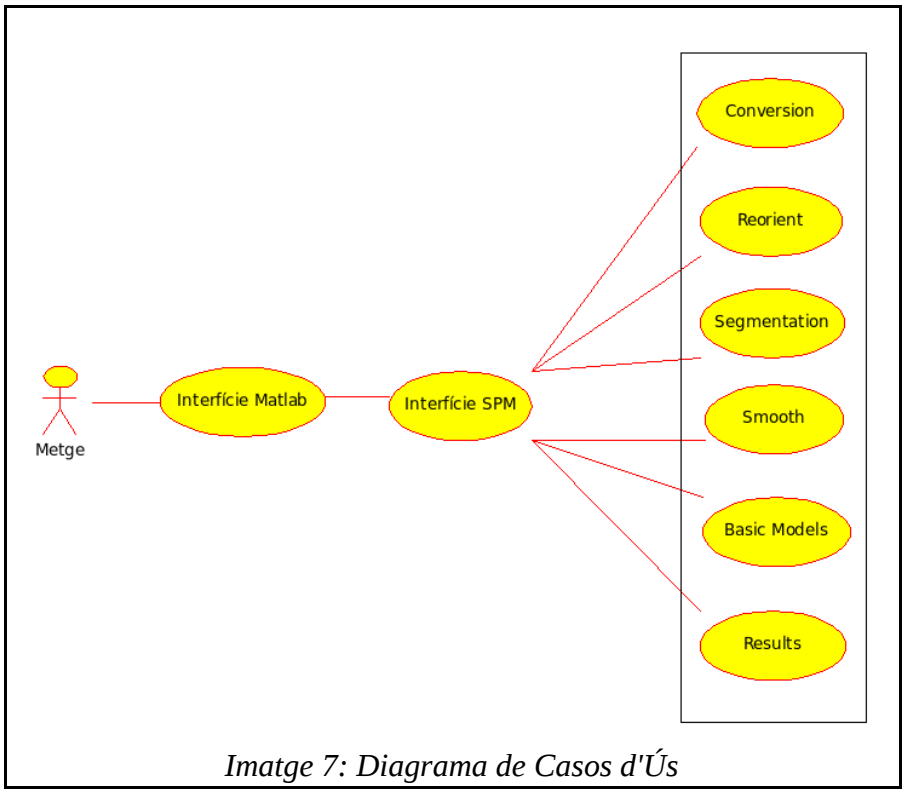
Taula 5: Procés per alinear les imatges

4.2 Anàlisis del software SPM

Durant l'estudi del software SPM s'ha analitzat el comportament de l'aplicatiu pel procés VBM definit anteriorment. Els següents diagrames següents i els adjuntats a A*nnexe I* reflexa l'anàlisi fet sobre el procés VBM.

4.2.1 Casos d'ús

Els casos d'ús s'especifiquen mitjançant la definició de dos elements, el diagrama de casos d'ús i les especificacions de cada cas d'ús. La definició està definida dins del model UML (Unified Modeling Language). Aquesta eina de modelització ens permet definir els rols, les funcionalitats i el comportaments relacionats amb el software. En el nostre cas representem la visió del paquet SPM, però limitada a l'abast del procés de VBM desenvolupat. El diagrama de casos d'ús, *imatge 7,* es defineix com un diagrama comportamental on es mostren gràficament els rols d'usuaris de l'aplicació (actors), les funcionalitat (casos d'us) i les seves interrelacions.



Imatge 7: Diagrama de Casos d'Ús

4.2.2 Anàlisi de les funcions principals

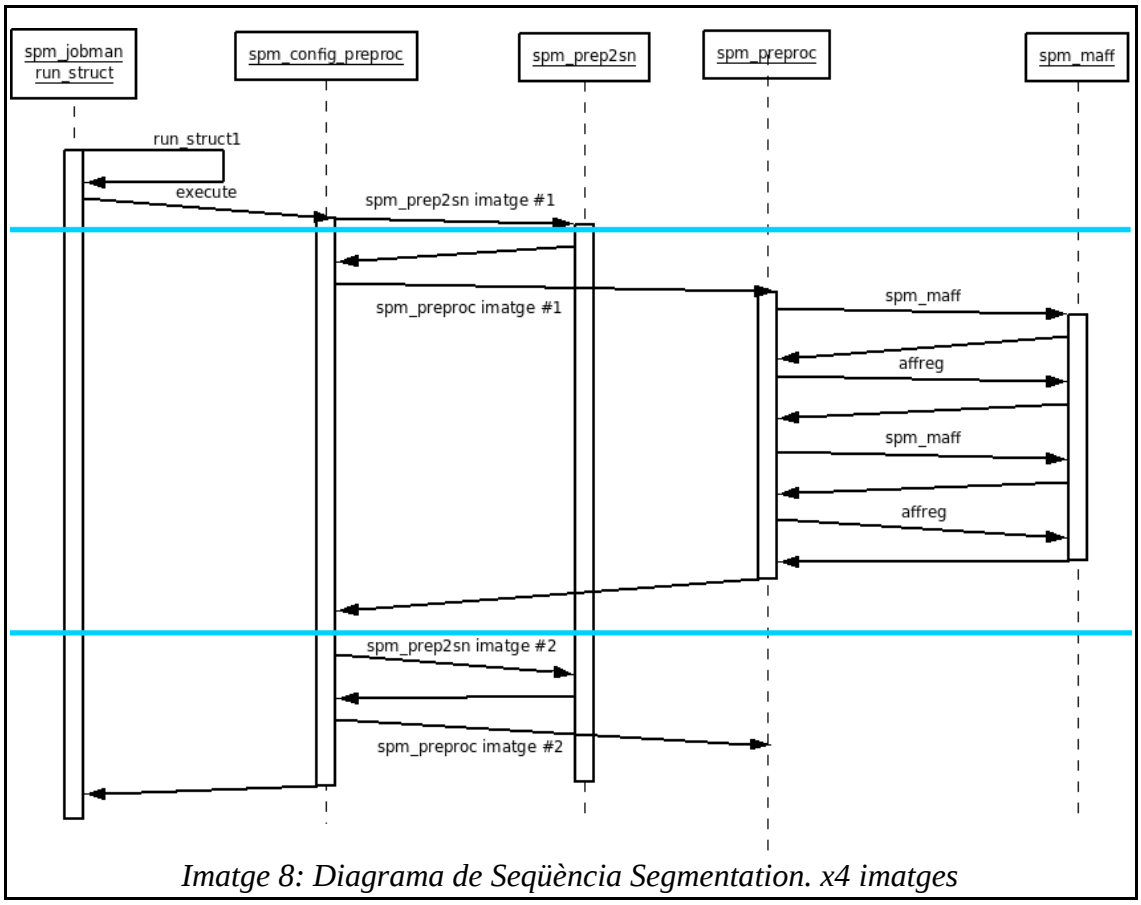
Es detallen a continuació els fils d'execució de les funcions principals que intervenen en els processos de teleprocessament d'imatges amb SPM. Els diagrames de seqüència s'ha extret a través de l'estudi del codi font i del sistema de Debug de Matlab.

La funció principal per tots els processos executats des de la interfície gràfica està ubicada al fitxer *spm_jobman.m*. Dels processos descrits en el procés de VBM s'han escollit els tres processos que han tingut el temps d'execució més alt: Segmentation, Smooth i Basic Models. Per realitzar els càlculs de temps s'ha seguit el procediment VBM amb 4 imatges RM (ressonància magnètica) per tal d'analitzar la relació de temps entre una i varies imatges. S'ha desenvolupat en Matlab una funció que ens permet generar un fitxer de Log per registrar el temps d'execució de cada funció. Amb els resultats he obtingut les funcions crítiques de cada procés.

Degut a que les característiques tècniques d'un equip poden provocar diferències en els temps d'execució del diferents processos, s'han fet servir un portàtil i una estació de treball per tal de comparar aquests temps d'execució *HW:1* i *HW:03*.

4.2.2.1 Procés de Segmentació

Diagrama de seqüència Es pot veure en la *imatge 8* la seqüenciació per una imatge i, mitjançant una línia blava, s'indica el punt on continua l'execució per la següent imatge, ja que aquest fil de seqüència es repeteix cada vegada per cada imatge de RM (ressonància magnètica).



Imatge 8: Diagrama de Seqüència Segmentation. x4 imatges

Temps d'execució El temps d'execució, *taula 6*, pel procés Segmentation per 4 imatges. Com és previsible la màquina XPS ha de tenir uns temps d'execució menors. Es pot veure que els temps de processament de cada una de la imatges és variable. La relació dels temps entre un ordinador i l'altre és de l'ordre de 5 vegades més ràpid. El temps total d'execució està definit per la funció principal spm_jobman amb un total de 8,08 minuts per *HW:01* i 37,53 minuts pel *HW:03*.

Arxiu	**Funció**	**Imatge**	**Temps DELL XPS HW:01**		**Temps Thosiba Lap HW:03**	
			segons	**minuts**	**segons**	**minuts**
spm_jobman.m	*run_struct*		484,5	8,08	2251,72	37,53
spm_jobman.m	*run_struct1*		484,49	8,07	2251,72	37,53
spm_config_preproc.m	*execute*		484,49	8,07	2251,69	37,53
spm_prep2sn.m	*spm_prep2sn*	1	21,22	0,35	66,67	1,11
spm_preproc.m	*spm_preproc*	1	88,31	1,47	441,55	7,36
spm_maff.m	*spm_maff #01*	1	1,36	0,02	11,47	0,19
spm_maff.m	*affreg #01*	1	1,36	0,02	10,61	0,18
spm_maff.m	*spm_maff #02*	1	6,48	0,11	51,81	0,86
spm_maff.m	*affreg #02*	1	6,48	0,11	50,98	0,85
spm_filepart.m	*spm_fileparts*	1	0	0	0	0
spm_prep2sn.m	*spm_prep2sn*	2	19,85	0,33	65,59	1,09
spm_preproc.m	*spm_preproc*	2	54,37	0,91	271,36	4,52
spm_maff.m	*spm_maff #01*	2	1,1	0,02	6,91	0,12
spm_maff.m	*affreg #01*	2	0,83	0,01	6,11	0,1
spm_maff.m	*spm_maff #02*	2	6,1	0,1	45,39	0,76
spm_maff.m	*affreg #02*	2	5,82	0,1	44,58	0,74
spm_filepart.m	*spm_fileparts*	2	0	0	0	0
spm_prep2sn.m	*spm_prep2sn*	3	22,63	0,38	73,09	1,22
spm_preproc.m	*spm_preproc*	3	72,63	1,21	370	6,17
spm_maff.m	*spm_maff #01*	3	2,3	0,04	16,33	0,27
spm_maff.m	*affreg #01*	3	2,01	0,03	15,45	0,26
spm_maff.m	*spm_maff #02*	3	6,75	0,11	50,66	0,84
spm_maff.m	*affreg #02*	3	6,46	0,11	49,77	0,83
spm_filepart.m	*spm_fileparts*	3	0	0	0	0
spm_prep2sn.m	*spm_prep2sn*	4	20,28	0,34	68,11	1,14
spm_preproc.m	*spm_preproc*	4	112,1	1,87	568,55	9,48
spm_maff.m	*spm_maff #01*	4	4,09	0,07	30,13	0,5
spm_maff.m	*affreg #01*	4	3,81	0,06	29,27	0,49
spm_maff.m	*spm_maff #02*	4	7,2	0,12	53,08	0,88
spm_maff.m	*affreg #02*	4	6,91	0,12	52,19	0,87
spm_filepart.m	*spm_fileparts*	4	0	0	0	0

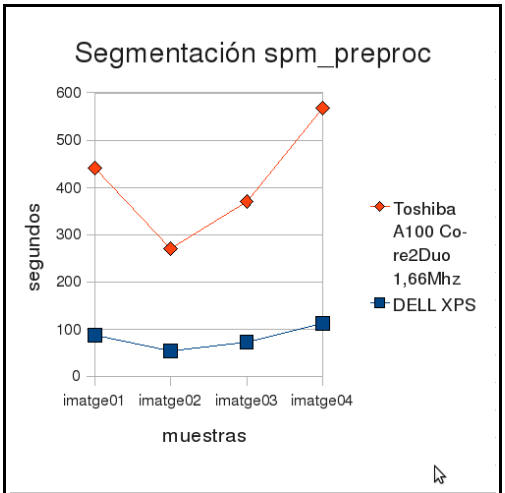
Taula 6: temps d'execució procés VBM en SPM

Els temps de la taula serviran per comparar-los amb els temps del model CUDA amb GPU. La funció crítica detectada durant la execució és **spm_preproc**. S'ha estudiat el codi que implementa hi s'hi veuen parts de codi, com per exemple bucles, que consumeixen gran part del temps. Es considera que aquesta funció pot ser paral·lelitzable, i serà una de les principals funcions a modificar utilitzant CUDA.

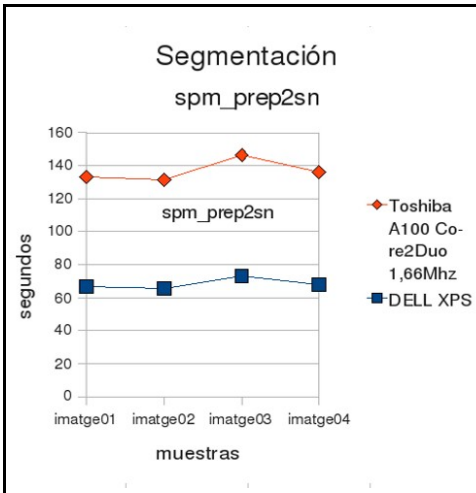
4.2.3 Conclusions

Després de veure els temps d'execució es considera que s'ha d'enfocar l'estudi de paral·lelització en el procés Segmentation. Aquest procés necessita un temps d'execució elevat, com es pot veure en la *taula 6*, on ha trigat 8 minuts pel DELL XPS i 40 minuts amb el portàtil Toshiba, que resulta d'un factor x5 entre un i altre hardware. S'ha comprovat que per les dues plataformes hardware el temps té una relació directa amb la màquina que l'executa com es veu a les *imatge 9* i *imatge 10*, on podem apreciar la similitud de la corba entre els temps de cada màquina. Calculada per cada funció també es compleix el mateix factor x5 entre els diferents maquinaris.

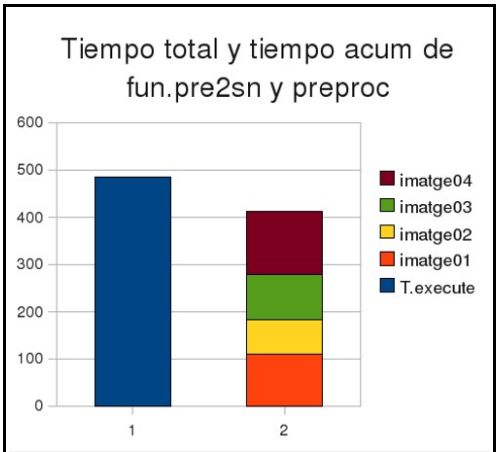
Aquest factor només ens indica la diferència de potència entre una màquina i l'altre recau únicament en el hardware, i per tant que segueix una linealitat. Per tant si es milloren els components, també es millorarà el temps d'execució. El temps d'execució està fortament lligat a les imatges. Per imatges diferents en la mateixa plataforma tenim temps d'execució diferents, però la mateixa imatge en màquines diferents es comporta linealment. Amb això es vol destacar que el temps d'execució ve determinat per les característiques intrínseques de les imatges. Es pot veure a *imatge 11* que la **imatge RM 2**, triga la meitat de temps que la **imatge RM1** o **imatge RM4**.



Imatge 9: a) Temps d'execució spm_preproc x4 imatges



Imatge 10: b) Temps d'execució spm_prep2sn x4 imatges



Imatge 11: c) Segmentation: Temps total i desglossat per imatge (preproc + prep2sn)

Taula 7: Temps execució SPM per procés de segmentació

```
for subit=1:40,
    oll  = ll;
    mom0 = zeros(K,1)+tiny;
    mom1 = zeros(K,1);
    mom2 = zeros(K,1);
    mgm  = zeros(Kb,1);
    ll   = llr+llrb;
    for z=1:length(z0),
        if ~buf(z).nm, continue; end;
        bf  = double(buf(z).bf);
        cr  = double(buf(z).f).*bf;
        q   =  zeros(buf(z).nm,K);
        b   =  zeros(buf(z).nm,Kb);
        s   =  zeros(buf(z).nm,1)+tiny;
        for k1=1:Kb,
            pr      = double(buf(z).dat(:,k1));
            b(:,k1) = pr;
            s       = s + pr*sum(mg(lkp==k1));
        end;
        for k1=1:Kb,
            b(:,k1) = b(:,k1)./s;
        end;
        mgm = mgm + sum(b,1)';
        for k=1:K,
  q(:,k) = mg(k)*b(:,lkp(k)) .* exp((cr-mn(k)).^2/(-2*vr(k)))/sqrt(2*pi*vr(k));
        end;
        sq = sum(q,2)+tiny;
        ll = ll + sum(log(sq));
        for k=1:K, % Moments
            p1      = q(:,k)./sq; mom0(k) = mom0(k) + sum(p1(:));
            p1      = p1.*cr;     mom1(k) = mom1(k) + sum(p1(:));
            p1      = p1.*cr;     mom2(k) = mom2(k) + sum(p1(:));
        end;
    end;
    % Mixing proportions, Means and Variances
    for k=1:K,
        mg(k) = (mom0(k)+eps)/(mgm(lkp(k))+eps);
        mn(k) = mom1(k)/(mom0(k)+eps);
        vr(k) =(mom2(k)-mom1(k)*mom1(k)/mom0(k)+1e6*eps)/(mom0(k)+eps);
        vr(k) = max(vr(k),eps);
    end;
    if subit>1 || (iter>1 && ~finalit),
        spm_chi2_plot('Set',ll);
    end;
    if finalit, fprintf('Mix: %g\n',ll); end;
    if subit == 1,
        ooll = ll;
    elseif (ll-oll)<tol1*nm,
        % Improvement is small, so go to next step
        break;
    end;
end;
```

Taula 8: Codi segmentació SPM

Podem dir que la linealitat està determinada pel software i les dades, i per tant es pot aplicar una millora sobre el codi estudiat. El procés Segmentation està limitat pel coll d'ampolla que suposa la funció spm_preproc. Després d'analitzar el seu contingut, *taula 8*, es veu que està composat per bucles aniuats. Aquesta estructura ens beneficia, de la forma que, es poden executar diverses etapes del bucle en diferents threads de la GPU.

4.3 Anàlisis del procés de Segmentació

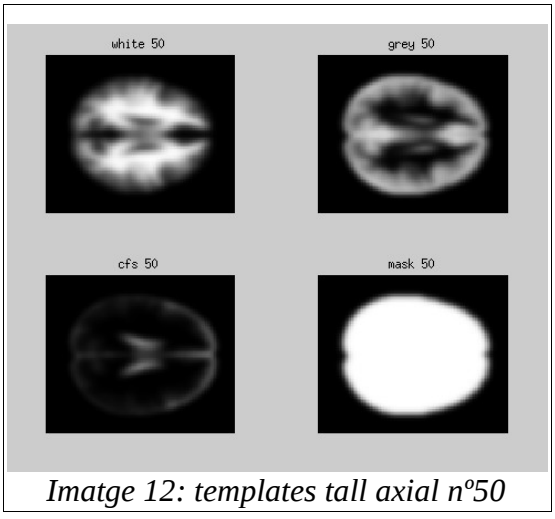
El procés de segmentació consisteix en separar els diferents tipus de teixits del cervell en teixit blanc, teixit gris i líquid cefaloraquidi, d'una imatge adquirida amb una RM (ressonància magnètica).

Per desenvolupar el procés de segmentació hem seguit el model descrit per *[John Ashburner*], que pertany al grup de desenvolupadors *The Wellcome Dept. Of Imaging Neuroscience*, creadors del paquet SPM. En el seu estudi *Humain Brain Function* descriu els processos de tractament d'imatges. El capítol 5, *Image Segmentation* tracta detalladament el procés de segmentació, sobre que el que ens hem basat per crear el prototip inicial de segmentació. El procés defineix un model de segmentació amb correcció de la no uniformitat de la intensitat (que pot aparèixer en l'adquisició de la imatge RM), per simplificar el procediment hem agafat el procés bàsic de segmentació, sense correcció de la no uniformitat de la intensitat.

El model assumeix que la imatge d'adquisició RM conté diferents tipus de teixits (clústers) al qual pot pertànyer un vòxel, les probabilitats a priori de que el vòxel pertanyi a un teixit o un altre, s'extreuen dels templates *[T1 template]*. Aquest template elaborat pel NMI (Neurologic Montreal Institute) com a part del ICBM (International Consortium for Human Brain Mapping). La plantilla ha estat desenvolupada en el projecte NIH P-20 (investigador principal John Mazziotta), que deriva de scans de 152 subjectes joves i sense cap malaltia diagnosticada, el grup és de 66 dones i 86 homes, 129 dretans, 14 esquerrans i 9 sense definir, edats compreses entre 18 i 44 anys, amb una mitjana d'edat de 25, les imatges d'adquisició tenen una normalització dins el mateix espai

estereotàxic fent servir 9 paràmetres (3 translacions, 3 rotacions i 3 zooms ortogonals) de transformació afí. Contenen valors entre 0 i 1, que representen la probabilitat a priori de pertànyer als clústers de GM, WM o CSF. També s'ha aplicat un filtre de suavitzat Gaussià full width at half maximum de 8mm. Les imatges estan emmagatzemades en format Nifti.

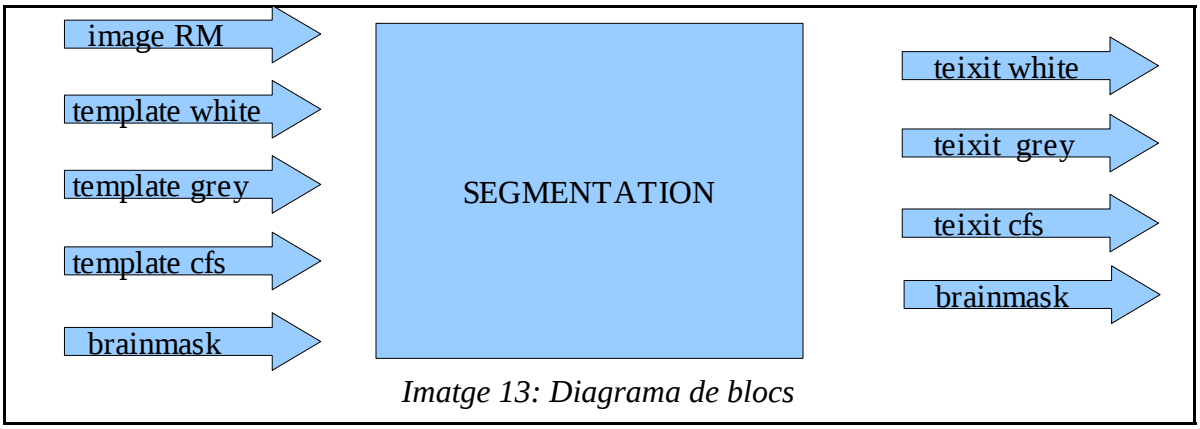
Per carregar les imatges en el prototip Matlab s'ha fet servir la llibreria MRI_TOOLBOX (Jimmy Shen, per CNSP de la universitat Flinders, Australia). La funció utilitzada és load_nifti, que carrega en el workspace de Matlab una estructura amb el format Nifti de la imatge emmagatzemada. Es pot extreure d'aquesta estructura la matriu amb els valors de l'adquisició. Per definició del format Nifti, les variables de la matriu són de tipus UINT8, i en la funció load_nifti de la imatge es realitza una transformació sform i qform que tenen per objectiu recomposar la imatge original aplicant informació de la capçalera per fer transformacions geomètriques, escalatge de la intensitat dels vòxels,etc. El resultat final de la adquisició de la imatge és una matriu tridimensional amb variables de tipus single. La dimensió dels templates és de 91x109x91 vòxels. La imatge d'adquisició a segmentar s'ha d'ajustar al mateix espai estereotàxic i dimensionalitat dels templates, i per adaptar-la fem servir la opció Normalize del paquet SPM. La *imatge 12* mostra els templates en el tall nº 50.



Imatge 12: templates tall axial nº50

4.3.1 Diagrama de Mòduls

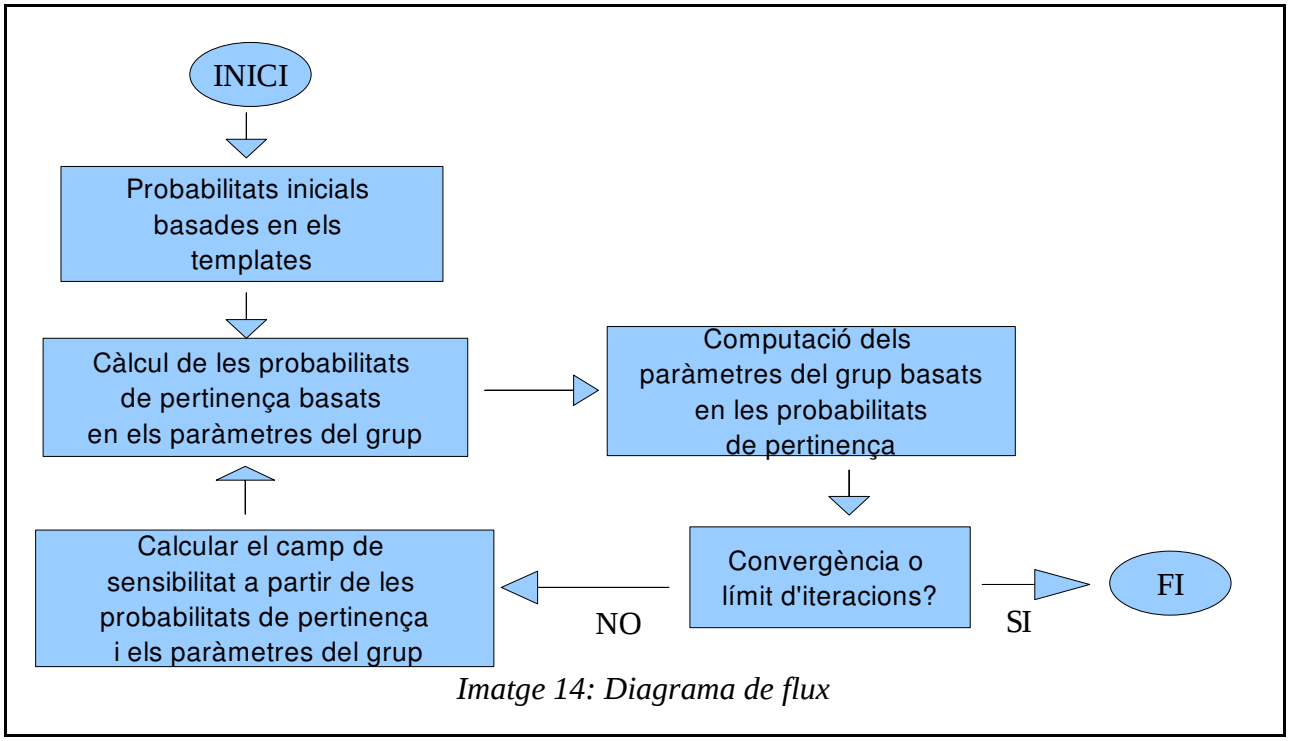
El procés de segmentació es pot veure com una procés que rep com a paràmetres la imatge RM a segmentar, els templates de teixit blanc, gris, líquid cefaloraquidi i la màscara. La màscara és la part de la imatge que no és teixit i que podem extreure de la diferència d'una matriu de ones menys els valors dels altres templates. El valor que retorna el procés és un mapa de probabilitats per cada clúster o teixit amb valores entre 0 i 1, que correspon a la probabilitat per cada vòxel de la imatge d'entrada de pertànyer a un o altre teixit. La representació del diagrama és la següent:



Imatge 13: Diagrama de blocs

4.3.2 Diagrama de Flux

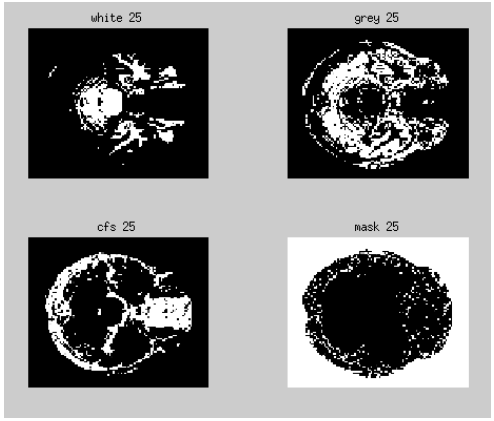
L'algorisme de segmentació simplificat, segueix el flux representat en la *imatge 14*.



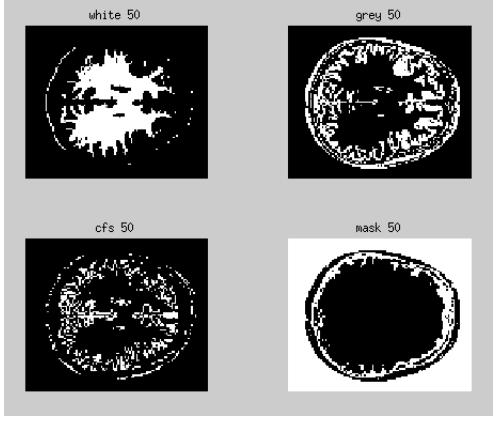
Imatge 14: Diagrama de flux

La convergència definida en el document s'esdevé quan el càlcul de l'error per una iteració respecte al càlcul d'error de la iteració anterior es inferior a un llindar (per exemple 0). Com el comportament de cada tall depèn de la naturalesa de la imatge, i com CUDA no suporta completament l'estàndard IEEE 754 de operacions en punt flotant, aquesta convergència no és igual per les versions C i CUDA, per tant s'agafa un altre criteri de convergència. En la implementació del model s'ha agafat com final de procés el llindar de 200 iteracions per cada tall.

El prototip inicial s'ha desenvolupat en Matlab, i basat en aquest prototip inicial s'ha fet la traducció al llenguatge C. Amb aquesta primera versió s'ha fet un càlcul del temps d'execució. Després s'ha agafat el prototip C, com a base per crear les diferents versions de codi paral·lelitzat executat en GPU. Al desenvolupar el codi sobre Matlab, els mapes de probabilitats finals queden com a variables dins l'espai de treball. Es poden visualitzar amb la comanda imshow, que pertany al toolbox gràfic de Matlab, i fer comparatives dels valors obtinguts entre les versions Matlab, C i CUDA. Es presenta a continuació, *imatge 15* i *imatge 16,* les imatges resultants de la segmentació corresponents als talls 25 i 50.



Imatge 15: Segmentació del tall 25

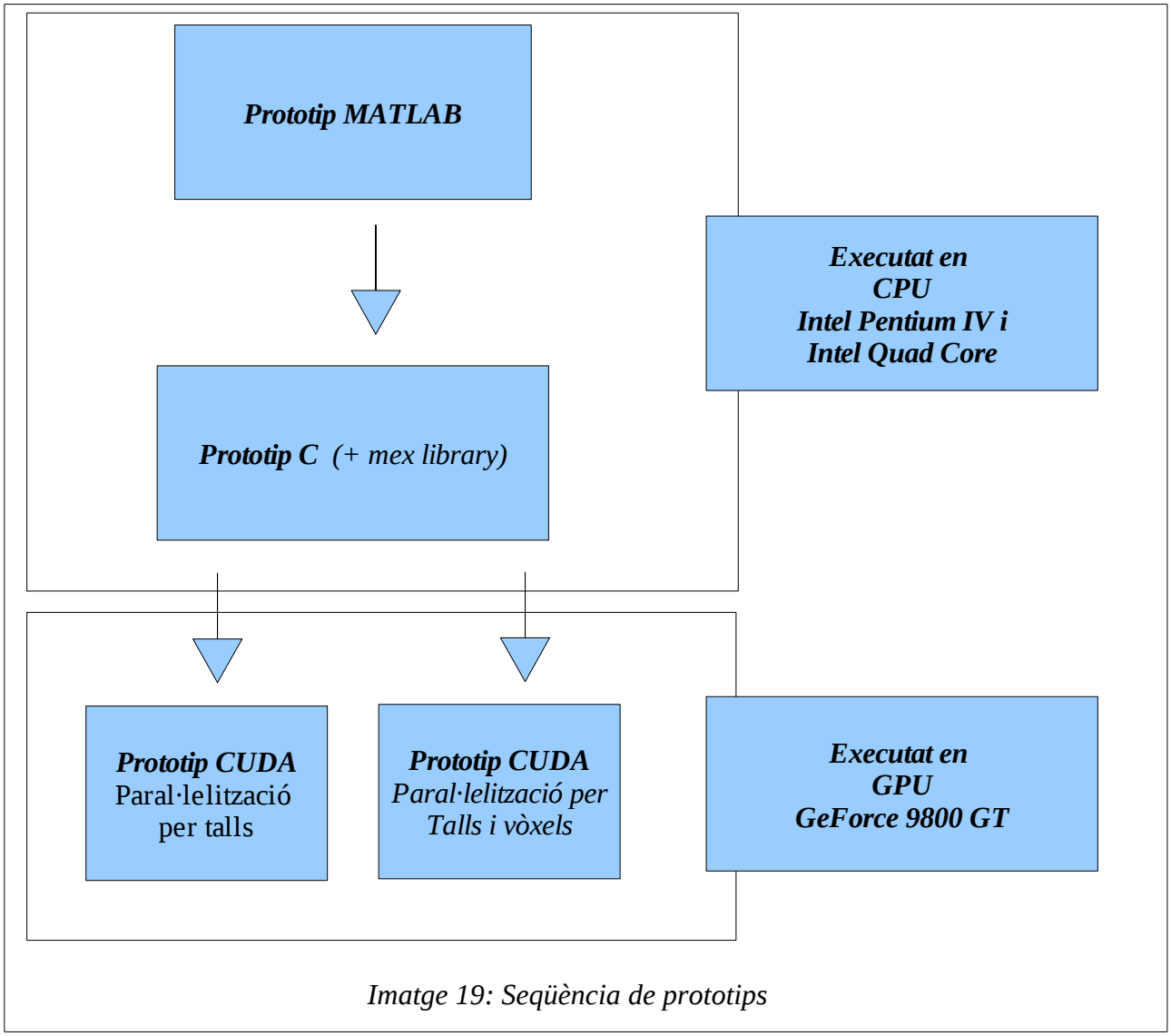


Imatge 16: Segmentació del tall 50

L'algorisme de segmentació té independència per cada tall, per tant es pot executar el procés per cada tall de forma paral·lela.

5 Implementació

Després de l'estudi del paquet VBM, es va fer una versió en C sobre la funció de segmentació spm_preproc del paquet SPM, en la seva part critica, però finalment es va decidir implementar l'algorisme de segmentació basat en el document [*John Ashburner*], per fer mes flexible la manipulació de les dades. Per tant es va desenvolupar el prototip base Matlab i sobre aquest les versions de C. La fase d'implementació ha necessitat versions successives del mateix software però aplicat a plataformes diferents i configuracions variables per realitzar els càlculs de temps d'execució.



Imatge 19: Seqüència de prototips

Per desenvolupar el prototip C s'han fet servir les llibreries **mex** , que permeten executar funcions C sobre l'entorn de Matlab. Les variables del workspace es passen com a paràmetres a la funció principal **mexFunction** sobre un arxiu amb extensió **.mex** i que utilitza el llenguatge C. Per compilar el codi haurem de tenir ben configurat el compilador **mex** (mex –setup), que es fà servir des de la línia de comandes de Matlab.

Amb la versió C estable s'ha desenvolupat diverses optimitzacions amb CUDA. En el transcurs d'aquesta fase s'ha pogut assistir a seminaris via web d'introducció al món de la programació amb CUDA, d'on s'ha pogut extreure documentació i exemples pel desenvolupament software, *[Webinar Jacket], [Webinar GPU], [Webinar Performance CUDA].*

5.1.1 Gestió dels recursos de GPU

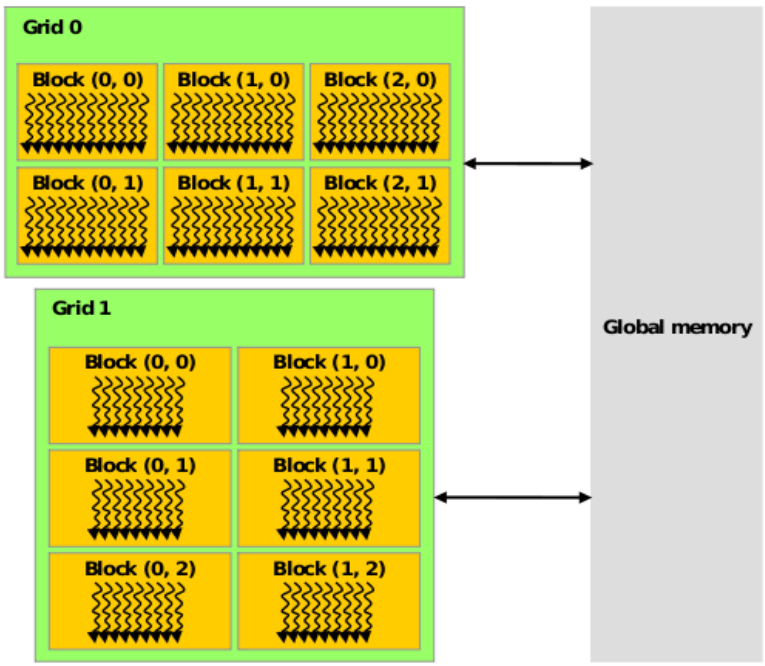
Quan treballem amb CUDA primer hem de diferenciar entre el host (memòria principal i CPU) i el device (memòria gràfica i GPU). La GPU nomes pot accedir a les dades del dispositiu i per tant les dades s'han de copiar de la memòria principal a la memoria del dispositiu. Per accedir a la memòria de la GPU, CUDA proporciona unes funcion per reservar espai (cudaMalloc), i per copiar informació entre el host-device (cudaMemcpy)de forma bidireccional.

A l'hora de desenvolupar el codi amb CUDA existeix una definició separada del codi executat en CPU i el codi executat en GPU, *imatge 17*. Les funcions GPU s'anomenen kernel, i llencen N threads que s'executen de forma paral·lela. En la crida a la funció es defineix l'estructura dels threads mitjançant la sintaxis <<<Grid, Bloc>>> on es pot configurar el tamany i la dimensió dels Grids i Blocs.

```
__global__ void vecAdd(float* A, float* B, float* C)
{
    int i = threadIdx.x;
    C[i] = A[i] + B[i];
}

int main()
{
    // Kernel invocation
    vecAdd<<<1, N>>>(A, B, C);
}
```

Imatge 17: Exemple de crida a kernel CUDA

La definició de la dimensionalitat i tamany dels Grid i Blocs ens permet accedir a les dades de forma que respectem la seva estructura natural, de forma que si les dades representen una matriu bidimensional, una definició d'un kernel de 2 dimensions, ens permet treballar al kernel per files i columnes. Existeix però, una limitació ja que els blocs poden definir un màxim de 3 dimensions i el Grid fins a 2. En el temps d'execució del kernel el processador executara un warp (32 threads d'un block) de forma simultània. En la definició del kernel també existeix la limitació de 512 threads per bloc i un màxim de 65536 * 65536 blocs. La configuració del kernel resulta important a l'hora d'accedir a l'identificador únic de cada thread, ja que es poden donar col·lisions en l'accés a la memòria si dos threads proven d'accedir a la mateixa posició de memòria, amb el risc d'inestabilitats. Els kernels s'executen paral·lelament a la CPU i també s'ha de controlar les col·lisions en aquest nivell, ja que dos kernels poden modificar dades de la memòria del dispositiu , però CUDA proporciona un punt de sincronisme entre funcions que es la funció *cudaThreadSynchronize*, que atura l'execució d'un kernel fins a l'acabament de l'anterior. Aquestes col·lisions les ha de detectar el programador per evitar una execució no controlada, ja que al executar-se en paral·lel, el dispositiu gestiona la llista de cues d'execució segons la disponibilitat dels recursos hardware. La *imatge 18* mostra l'estructura de dos kernels, i com accedeixen a la mateixa memòria, per tant s'hauran de controlar les dependències de dades WAW, (write after write) RAW(read after write).

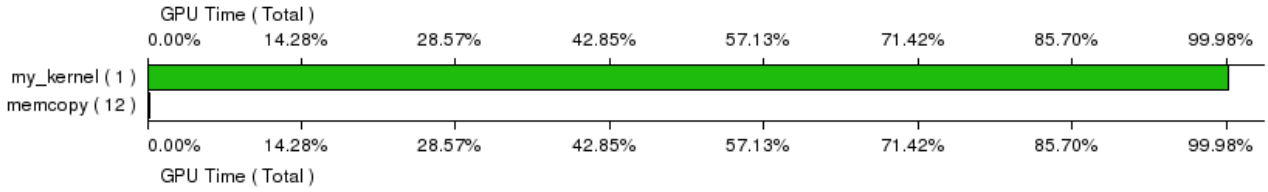


Imatge 18: Accés compartit a la memòria del dispositiu

5.2 Versió CUDA 1 (paral·lelitzat per talls)

La primera versió CUDA aprofita la independència de computació dels talls en l'algorisme de segmentació, i executa un kernel, amb un únic Grid i un bloc de 91 threads. Aquesta execució ha estat la mes ràpida d'implementar, ja que s'ha agafat el codi de segmentació en C i s'ha pogut adaptar fàcilment a una funció de tipus kernel CUDA, però per contra no aprofita al màxim els recursos del device. Els speedups esperats a priori es de 2,8x respecte a la versió C. Això es perquè sabem que executa 91 threads paral·lels, i el tamany màxim del warp es 32, que dona el factor de 2,8x. S'ha de tenir en compte que en el càlcul dels temps empírics existeix la variable temporal en la copia de les dades entre el host i el dispositiu, que suposa una de les operacions mes lentes, però en el nostre cas, no ha ocupat més que un 0,01% del temps, degut al poc pes de les imatges.

En la següent imatge podem veure la distribució proporcional del kernel respecte a la transferència de dades a la GPU.



Imatge 20: Distribució de temps dels processos GPU

method	**#calls**	**GPU usec**	**CPU usec**	**%GPU time**
my_kernel	1	1,46E+008	1,46E+008	99,98
memcopy	12	24674,8	28058	0,01

Taula 9: Distribució del temps per cada funció de la versió CUDA 1

5.3 Versió CUDA 2 (paral·lelitzat per tall i per vòxel)

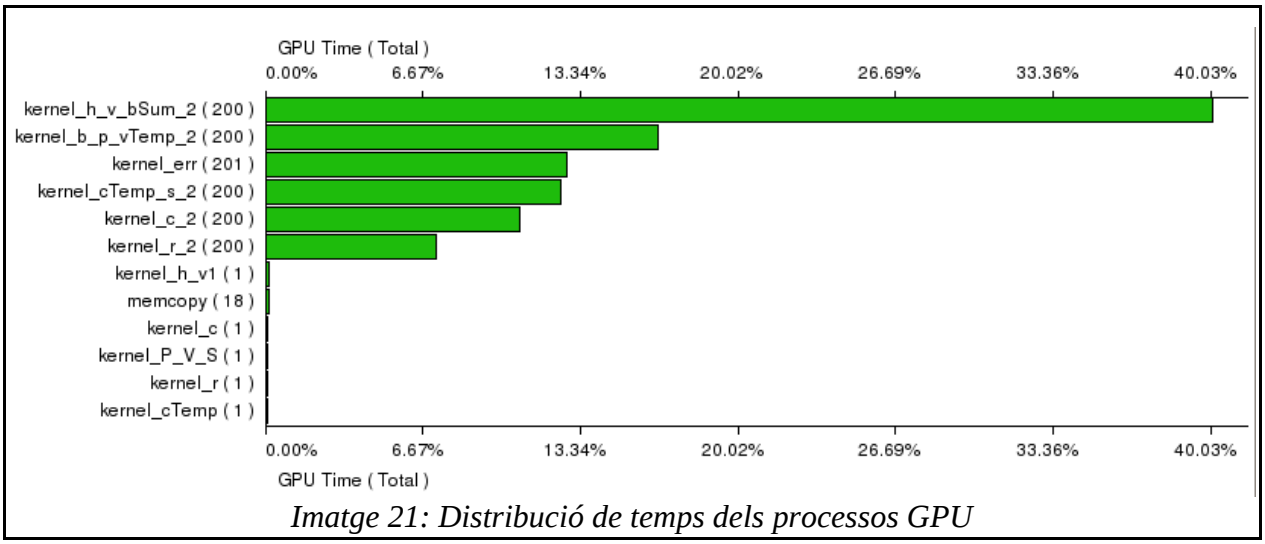
La segona optimització aprofita millor la configuració del kernel, i executa 11 kernels diferents amb dos tipus de configuracions. La primera configuració es per els codi amb dependència per talls amb kernels d'un Grid i un bloc de 91 threads. La segona configuració pel codi on no existeix dependència s'aprofita una execució paral·lela per cada vòxel amb un grid2D (91,109) i un bloc de 91 threads. Les característiques de l'algorisme demanen càlculs del sumatori d'un tall, on s'aplica el paral·lelisme per talls (per exemple la operació Matlab "hw = sum(pw)"), i en les operacions per vòxel es fa servir la configuració 2 (per exemple la operació "vw= vw1./hw;").

Una de les consideracions que es van tenir en compte, es la permanència de les dades al device entre les execucions dels diferents kernels, i l'accés a aquesta memòria únicament a l'inici i al final de l'algorisme, ja que la transmissió entre la memòries host i device es una de les operacions mes lentes. Per evitar les col·lisions a les dades del device després l'execucióun kernel es realitza un punt de sincronisme amb la funció *CudaThreadSynchronize,* ja que les dades calculades per un kernel son accedides pel següent.

```
// DEFINICIÓ DELS KERNELS

in.x = 91; in.y = 109, in.z = 91;

dim3 dimGridBig(in.x*in.y); dim3 dimBlockBig(in.z);

dim3 dimGrid(1); dim3 dimBlock(in.z);


// EXECUTA ELS KERNELS

// KERNEL AMB PARAL·LELISME PER VÒXELS

kernel_b_p_vTemp_2<<< dimGridBig,dimBlockBig>>>
(xD,yD,zD,bwD,bgD,bcD,bmD,pwD,pgD,pcD,pmD,vwTempD,vgTempD,vcTempD,vmTempD,rwD,rgD,rcD,rmD,swD,sgD,scD,smD,imatgeD);

cudaThreadSynchronize();          // sincronisme per evitar col·lisions en l'accés
                                  // a pwD,pgD,pcD,pmD, vars. utilitzades pel següent kernel

// KERNEL AMB PARAL·LELISME PER TALLS

kernel_h_v_bSum_2<<< dimGrid,dimBlock>>>
(xD,yD,zD,hwD,hgD,hcD,hmD,vwD,vgD,vcD,vmD,bwSumD,bgSumD,bcSumD,bmSumD,pwD,pgD,pcD,pmD,vwTempD,vgTempD,vcTempD,vmTempD,bwD,bgD,bcD,bmD);

cudaThreadSynchronize();
```

Taula 10: Codi d'exemple de la versió CUDA 2. Crides per diferents configuracions.

El la *imatge 21*, veiem la proporció dels temps consumits per cada procés GPU. El kernel kernel_h_v_bSum_2, executat 200 vegades, consumeix el 40% del temps total GPU.



Imatge 21: Distribució de temps dels processos GPU

La *taula 11* s'ha extret amb l'eïna *[CUDAProfiler]*, amb la qual podem avaluar el comportament de la funció, i detectar els processos a optimitzar en cas que sigui possible.

method	**#calls**	**GPU usecs**	**CPUsec**	**%GPU time**
kernel_h_v_bSum_2	200	1,73E+007	1,73E+007	40,03
kernel_b_p_vTemp_2	200	7,15E+006	7,15E+006	16,58
kernel_err	201	5,47E+006	5,47E+006	12,68
kernel_cTemp_s_2	200	5,38E+006	5,38E+006	12,47
kernel_c_2	200	4,61E+006	4,61E+006	10,69
kernel_r_2	200	3,09E+006	3,09E+006	7,17
kernel_h_v1	1	46849,6	46852,6	0,1
kernel_c	1	22992	22994	0,05
kernel_P_V_S	1	17858	17865	0,04
kernel_r	1	15548,3	15550,3	0,03
kernel_cTemp	1	12390,9	12393,9	0,02
memcopy	18	38896,3	43304	0,09

Taula 11: Distribució del temps per cada funció de la versió CUDA 2

5.4 Temps d'execució

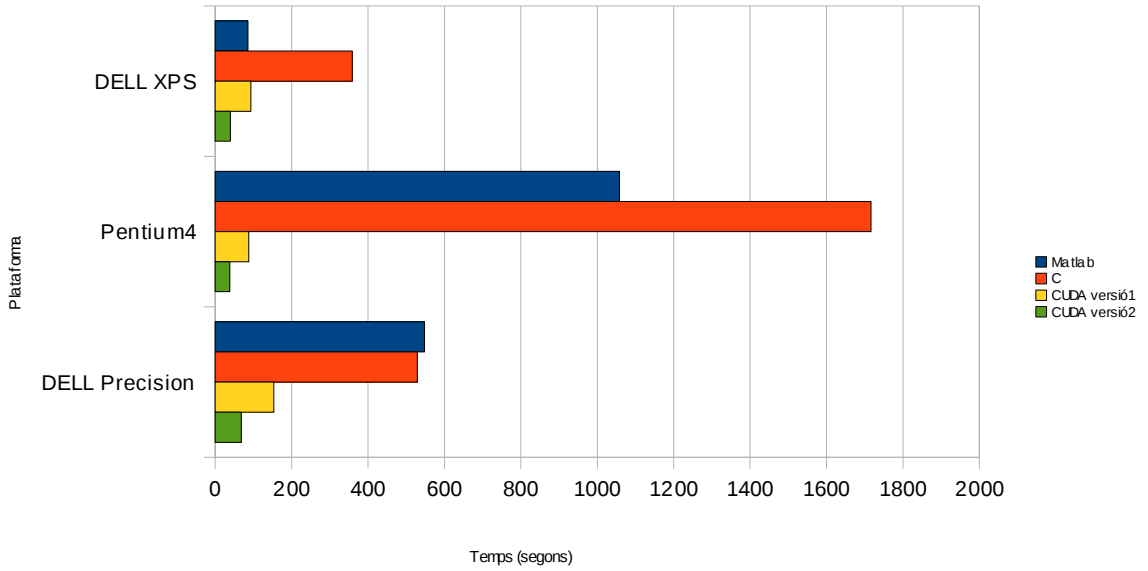
En la *taula 12* reflexem els temps d'execució sobre les plataformes hardware *HW:01*,*HW:02, HW:04 i HW:05*. Les plataforma *HW:01* té dues targetes gràfiques i s'ha pogut executar el codi des de l'entorn Matlab, amb interfície gràfica a la primera targeta gràfica, i els càlculs executats sobre la segona targeta gràfica. En l'entorn *HW:02* només disposa d'una targeta gràfica i s'ha executat la versió standalone, des de línia de comandes. L'entorn *HW:04* no disposa de suport CUDA i s'ha executat la versió de Matlab i C per fer una comparativa entre aquestes dues versions. A la columna d'speedup tenim dos valors, el primer és el guany respecte a la versió anterior i després el càlcul sobre la versió Matlab inicial.

	HW:01 DELL XPS geForce 9800GT		**HW:02 DELL Precision Quadro FX 1600M**		**HW:04 DELL Latitude GMA 950 (no CUDA)**		**HW:05 Pentium4 geForce9800GT**	
	minuts	speedup matlab i vers.ant.	minuts	speedup matlab i vers.ant.	minuts	speedup matlab i vers.ant.	minuts	speedup matlab i vers.ant.
Matlab	85,99	-	547,67	-	1394,36	-	1058,58	-
C	358,51	*0,24* *0,24*	528,81	*1,03* *1,03*	1040,66	*1,33* *1,33*	*1716,77*	*0,61* *0,61*
CUDA v1 talls	93,31	*0,92* *3,84*	153,38	*3,57* *3,44*	-	-	87,85	12,04 19,54
CUDA v2 vòxels	44,01	*1,95* *2,12*	68,57	*7,98* *2,23*	-	-	38,19	27,71 2,30
Speedup cuda v2 vs Matlab		***1,95***		***7,98***				**27,71**

Taula 12: Temps d'execució en plataformes hardware diferents

Segons la taula de temps obtinguts per les plataformes, podem veure que hem obtingut un guany respecte a la versió anterior, amb un màxim speedup de 27x, que correspon a la versió Matlab vs. CUDA versió 2. Cal destacar el temps obtingut per la versió Matlab sobre la plataforma DELL XPS *HW:01*, ja que ha donat un temps molt bó en aquesta primera versió. Al executar aquesta versió en plataformes diferents, veiem que aquest comportament és puntual, i creiem que es degut a l'aprofitament de Matlab dels 4 cores del processador mes la optimització dels càlculs per matrius del software Matlab. Tot i així en la CUDA versió 2 hem millorat el temps amb un speedup de 2x sobre aquesta versió Matlab.

Cal destacar en la *imatge 22* que el temps CUDA HW:01 i HW:05 amb la mateixa targeta gràfica obtenen el mateix temps. Per tant una màquina PentiumIV de categoria inferior i preu més assequible obté els mateixos resultats. Per contra el resultat de DELL Precision HW:02 amb una targeta de categoria inferior també obté molt bon resultat.



Imatge 22: Comparativa de temps. Les dues primeres amb GeForce 9800 GT i la màquina DELL Precision amb una Quadro FX 1600M

5.5 Proves unitàries

Expliquem algunes de les proves unitàries que s'han fet per analitzar els resultats obtinguts entre els diferents prototips realitzats. Una de les situacions que van aconsellar aprofundir en aquest estudi és degut a la diferència en el número d'iteracions que fan falta per convergir el procés de segmentació entre les versions de Matlab, C i CUDA. Al finalitzar aquestes proves, es va decidir agafar com a criteri de convergència, el llindar de 200 iteracions per cada tall. D'aquesta manera podem assegurar que les versions son comparables en quant al flux d'execució.

5.5.1 Proves d'arrodoniment

En realitzar un estudi del codi C, es va analitzar el nombre d'iteracions per cada tall i es va comparar amb els resultats obtinguts en la versió Matlab. Es va aprofundir aïllant un tall i comprovant els valors parcials. Es va veure que els valors obtinguts en les operacions amb les matrius donaven una petita diferència en els valors. En començar amb la primera operació que consisteix en el sumatori dels elements d'una matriu s'obtenia un valor diferent, i aquest error

s'anava propagant en l'algorisme que provocava que el criteri de convergència en error 0 no finalitzes en el mateix nombre d'iteracions.

Per la suma dels elements del tall 25, en el template CFS, s'obté en Matlab el valor 4.4191916e+02, i en C el valor 4.4191904e+02. En va revisar la documentació de Matlab i C i en principi els dos implementen l'estàndard IEEE per aritmètica en coma flotant (IEEE 754). Per tant el resultat hauria de ser el mateix. En la documentació es va veure que Matlab tracta les operacions amb matrius d'una forma concreta, fent una suma per columnes i després la suma dels resultats parcials, això genera una diferència respecte a la suma iterativa element per element dels valors d'una matriu amb bucles aniuats desenvolupada en C. Per tant, es va modificar el prototip Matlab per simular una suma dels elements de forma seqüencial, per substituir la funció sum() utilitzada per fer aquest càlcul.

```
filename = './imatges/csf.nii';
csf = load_nii(filename);

Aa= csf.img(:,:,25);                    % Aa = <91 x 109 single>
Bb= csf.img((24*91*109)+1:(25*91*109));% Bb = <1 x 9919 single>


resA1 = sum(sum(Aa))   % valor obtingut 4.4191916e+02
resB1 = sum(sum(Bb))   % valor obtingut 4.4191904e+02

A = Aa;                % A = <91 x 109 single>
B =reshape(Bb,91,109); % B = <91 x 109 single>


% Si realizamos la suma los valores ahora son iguales
resA2 = sum(sum(B)) % 4.4191916e+02
resB2 = sum(sum(A)) % 4.4191916e+02
```

Taula 13: Prova Unitària

En aquesta prova de test podem veure que si es tracta d'una matriu bidimensional (Aa), el resultat dona un resultat (resA1) , que es diferent (resB1) si la suma és d'un vector (Bb).

Les proves d'arrodoniment amb CUDA respecte la versió en C també donen diferents resultats, però en revisar la documentació de les notes de distribució, aquesta limitació ve indicada. Tal com indiquen, la precisió simple, els nombres desnormalitzats i els NaNs, no estan suportats, només dos IEEE modes d'arrodoniment estan suportats (chop and round-to-nearest even), la precisió de la instrucció divisió i arrel quadrada és lleugerament més petita que la precisió simple.

5.5.2 Càlcul d'error

Per analitzar els resultats de segmentació obtinguts entre les diferents versions, s'ha fet un petit estudi de les matrius de probabilitat obtingudes. Per realitzar les comparacions s'ha guardat les matrius resultants i s'han realitzat la comparativa en l'entorn Matlab. S'han agafat les matrius del clúster de la matèria blanca (WM).

Màxim error

Amb aquesta prova mostrem l'error màxim obtingut entre les versions. Es resten les matrius entre dues versions i es mostra el valor màxim, que correspon a la diferència màxima entre valors obtinguts. Hem de tenir en compte que els valors obtinguts varien entre 0 i 1.

La *taula 14* mostra el codi en llenguatge Matlab que ens mostra el màxim error entre les versions de c i cuda per la matriu que representa la matèria blanca.

```
clear
load varsC;
load varsCuda;
load varsM;
load varsCudaOp;
a = cSegw(:,:,:);
b = cudaSegw(:,:,:);
mat = abs(a - b);
maxerror = max(max(max(mat)));
```

Taula 14: Codi Matlab per obtenir el màxim error

En la *taula 15* podem veure que l'error més gran el trobem sempre en les versions de C respecte a les altres implementacions. En canvi CUDA i Matlab obtenen resultats molt semblants amb una diferència màxima de 0.0000810, que és un valor prou petit, com per dir que els resultats obtinguts entre aquestes versions és equivalent.

SEGMENT. WHITE	**CUDA v1**	**CUDA v2**	**c**	**m**
CUDA v1	-	0	0,61	8,10E-005
CUDA v2	0	-	0,61	8,10E-005
c	0,03	0,03	-	0,03
m	2,12E-004	2,12E-004	0,61	-

Taula 15: Màxim error per les diferents versions

Nombre d'errors

En la següent prova, es computa el nombre de valors diferents entre les versions.

```
s = find (a~=b & isnan(a)==0 & isnan(b)==0)
numerrors = size(s);
```

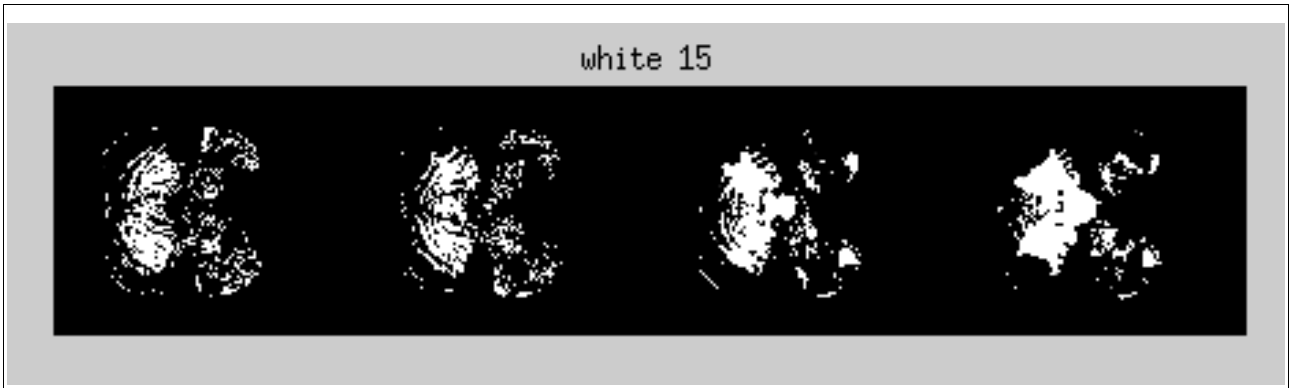
Si el número total d'elements de la imatge és 91x109x91 (902629 vòxels), la versió de CUDA obté 20.286 valors diferents (44,45%), però ,tal com diu l'apartat anterior, on la diferència màxima és de 0.0000810.

SEGMENT. WHITE	**CUDA v1**	**CUDA v2**	**c**	**m**
CUDA v1	-	0	21170	20286
CUDA v2	0	-	21170	20286
c	21170	21170	-	11276
m	20286	20286	11276	-

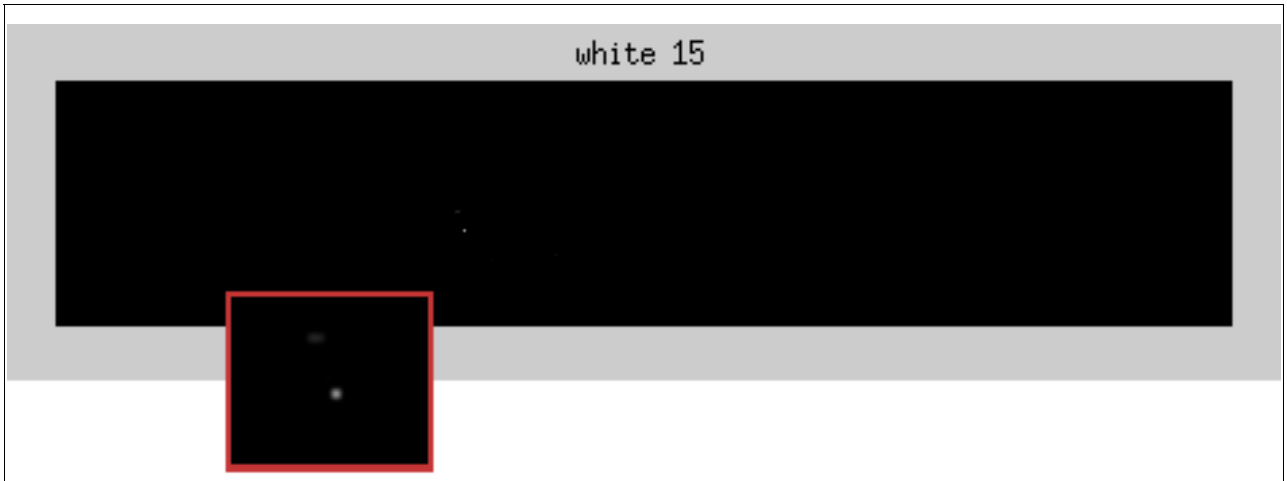
Taula 16: Nombre d'errors per les diferents versions

Segons les taules anteriors el pitjor resultat el veiem entre les versions de C i CUDA, on tenim 21.170 valors diferents (42.63%) on l'error màxim és de 0,61. En el pitjor dels casos, i agafant un classificador de màxima probabilitat per cada teixit, voldria dir que entre les versions de C i CUDA existeix una diferència de resultats del 43% per la classificació dels vòxels del teixit white, que és molt alta. Per això s'ha volgut veure la distribució de l'error en aquest cas.

El primer que s'ha fet es localitzar el tall on es troba el màxim error, que està situat al tall 15, *imatge 23* i *imatge 24*. A simple vista els resultats son similars entre les versions, però en el detall del tall 15 podem veure una petita taca que representa l'error màxim d'aquest clúster entre les versions C i CUDA.



Imatge 23: Segmentació matèria blanca, talls 14 - 17



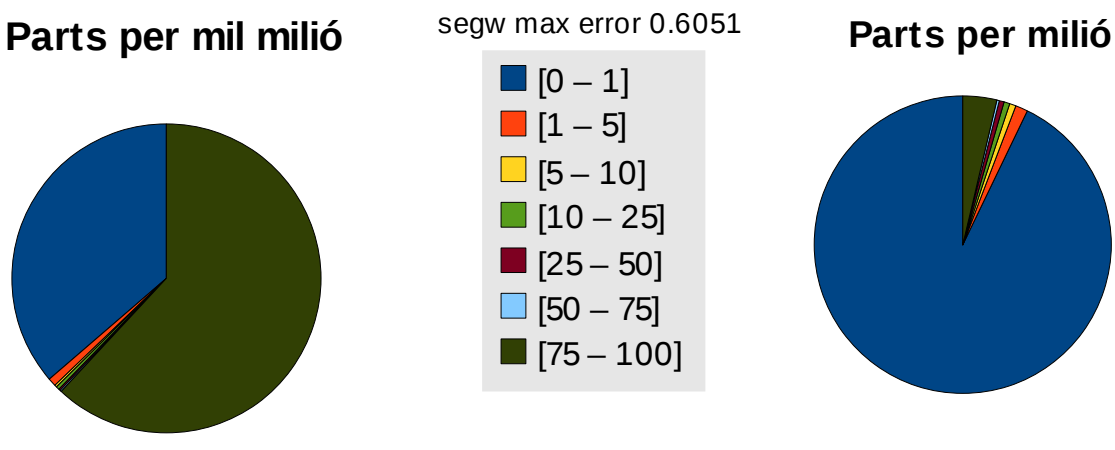
Imatge 24: Visualització de la diferència entre els resultats CUDA i C, amb un detall del valor màxim 0.61

Per veure la distribució dels errors respecte respecte les versions , podem veure en la *taula 17*, que el 99,94% dels errors estan per sota de l'1% . Per tant el 0.06% dels errors estan per sobre de 0.006051

intervals distribució	nº elements	percentatges	nº elements	parts de milió	nº elements	parts de mil milió
[0 – 1]	21158	99,94	19651	92,8	7686	36,31
[1 – 5]	7	0,03	285	1,35	181	0,85
[5 – 10]	1	0	147	0,69	52	0,25
[10 – 25]	3	0,01	136	0,64	68	0,32
[25 – 50]	0	0	108	0,51	42	0,2
[50 – 75]	0	0	61	0,29	34	0,16
[75 – 100]	1	0	782	3,69	13107	61,91
	21170	*100*	*21170*	*100*	*21170*	*100*

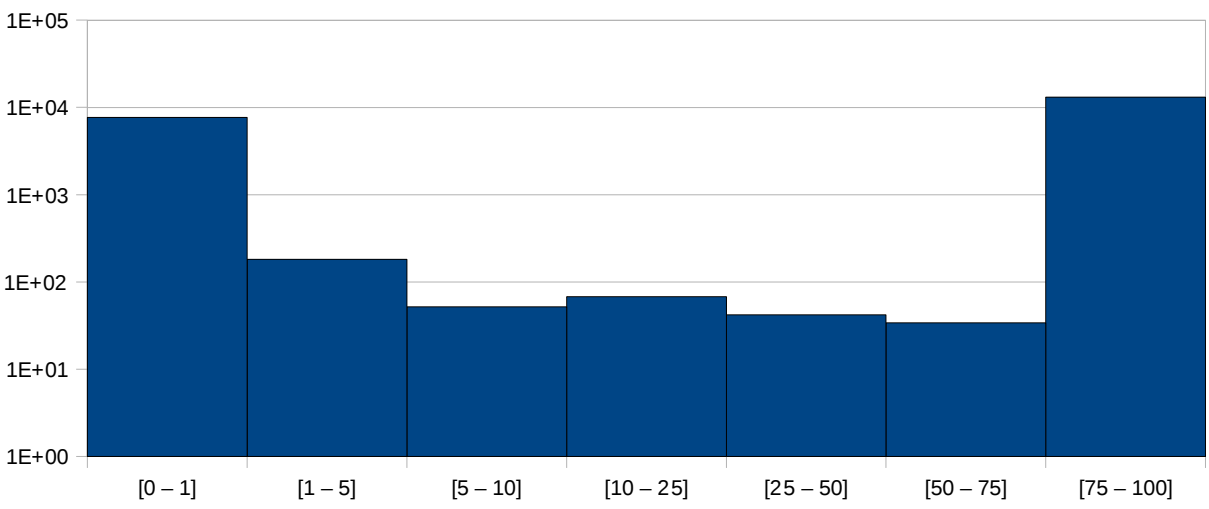
Taula 17: Intervals en percentatges, parts de milió i parts de mil milió sobre el màxim error

En la *imatge 25* reduïm els percentatges a parts de milió i part de mil milió, per veure la distribució de l'error per sota de 1%, i ens serveix per veure si l'error es prou gran com per ser apreciable i per tenir en consideració.



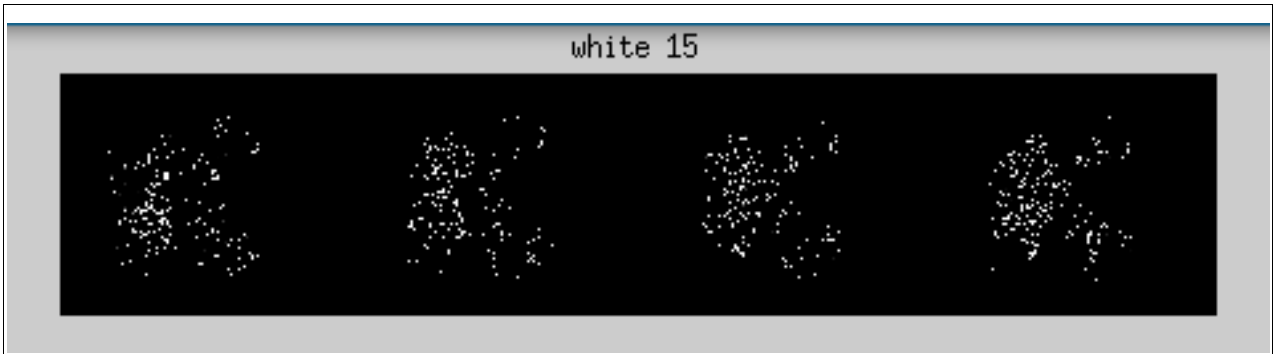
Imatge 25: Distribució de l'error

La *imatge 26*, no representa les proporcions en l'eix X, però ens mostra la distribució dels errors en ordre exponencial per parts de mil milió.



Imatge 26: Distribució de l'error en parts de mil milió

Per tant podem apreciar la distribució del error entre les versions C i CUDA multiplicant per la magnitud 10^9, *imatge 27*, i visualitzant en una representació bidimensional, que comparada amb la imatge incial podem veure que els errors es troben en àrees d'incertesa, i que es consideren prou petites com per ser imperceptibles. El màxim error és un valor puntual i no representa una constant sobre els resultats entre les versions., i encara que no podem evitar aquestes diferències, són prou petites per donar el resultat obtingut per bo.



Imatge 27: Diferències en la segmentació dels talls 14-17 teixit blanc

5.6 Problemes

Al llarg de la implementació han sorgit una sèrie de problemes que s'han anat resolent durant el transcurs de diverses fases. El primer problema que ha sorgit és amb la versió inicial amb C sobre la funció spm_preproc del paquets SPM, ja que es va implementar la funció però codificant un estat intermitg del procés que no resultava molt flexible a l'hora de programar, ja que es basa en mantenir les variables del workspace i inserir una crida a una funció C que substitueix a la part del codi crític de segmentació. Per això es va tornar a implementar des de l'inici l'algorisme sobre el document Ashbourner.

Un altre problema que es va descobrir sobre CUDA en la limitació d'execució d'un kernel a 5 segons. Després dels 5 segons el sistema operatiu dona un error sobre un procés inactiu Això es degut a que tant en Windows com en Linux, es dispara un Watchdog del controlador gràfic, que atura l'execució del codi, retornant un error. Una primera solució a aquest problema és la instal·lació d'una segona targeta gràfica a l'estació de treball, i a l'hora de llençar el kernel, executar-lo sobre la segona GPU, ja que d'aquesta manera la memòria de la GPU no comparteix recursos entre el codi GPGPU i la interfície gràfica, aquest mètode és necessari si volem llençar l'execució des de Matlab que requereix un entorn gràfic per ser executat. Una altra forma de solucionar el problema és executant el codi des de línia de comandes en una terminal, i per això es va crear una versió standalone, que no necessita l'entorn Matlab perquè carrega les imatges des de fitxer. Aquesta solució es va fer servir per fer les comparatives de temps amb el portatil DELL Precission *HW:02*, ja que no és l'altre entorn compatible amb CUDA, però que no permet la instal·lació d'una segona targeta gràfica.

El problema de l'arrodoniment va causar un problema important a l'hora de validar la versió C, ja que l'algorisme es basa en la convergència de l'error, i en l'execució C el nombre d'iteracions fins a la convergència no corresponia al nombre d'iteracions del prototip, per tant, com l'enfocament del projecte és fer un estudi dels temps d'execució, es va decidir equiparar els dos codis definint un nombre d'iteracions idèntic per les dos versions de codi. Finalment la versió realitza 200 iteracions per cada tall, ja que en l'estudi de les versions es va veure que normalment els talls convergeixen en menys de 20 0 iteracions, i ens permet garantir que les imatges resultants són vàlides.

6 Conclusions

S'ha aconseguit desenvolupar un procés de neuroimatge i s'ha obtingut un cert speedup fent servir la tecnologia GPGPU, a més s'ha acoseguit integrar dins l'entorn Matlab. També s'han desenvolupat versions standalone que han permès executar el codi en màquines amb una única targeta gràfica, degut a que l'execució des de l'entorn Matlab requereix un device i el càlcul de coprocessament un altre, i per tant s'han pogut fer comparatives en diferents plataformes.

Podem conclure que la tecnologia GPGPU s'adapta perfectament al tractament eficient d'imatges mèdiques. S'ha de avaluar que la millora obtinguda depèn de la naturalesa de l'algorisme (grau de paral·lelització) i del disseny d'un kernel que aprofiti al màxim els recursos GPU.

La desviació temporal respecte a la inicial està justificada amb els estudis d'anàlisi que han limitat l'abast del projecte.

En el desenvolupament d'aquest projecte s'ha estudiat el tractament de VBM (Voxel Based Morphometry) i després de l'estudi inicial s'ha centrat en la part amb més cost computacional d'aquest procés, el prototip generat correspon al mòdul de segmentació. Sobre aquest mòdul s'han fet els càlculs de temps i s'ha elaborat un informe amb els estudis comparatius per a cada plataforma i configuració.

En la implementació, primerament s'ha desenvolupat un prototip en llenguatge C equivalent a la part de la funció de segmentació de Matlab, però per simplificar l'estudi es va decidir tornar a programar des de l'inici el procés de segmentació. Els motius són principalment la comprensió de l'algorisme que s'estava desenvolupant, i aïllar el procediment de segmentació. Degut a que, en la funció Matlab d'SPM, a banda de fer la segmentació realitza algorismes de tractament de la imatge per la reducció de soroll mitjançant la correcció de la distribució no uniforme de la intensitat. Per la realització del prototip i l'estudi comparatiu ha estat necessari afegir aquesta fase

on s'ha tornar a desenvolupar un estudi de viabilitat, i uns documents d'anàlisi i disseny. Una vegada elaborat el prototip en Matlab del procés de segmentació, de forma independent a SPM, s'ha tornat a fer la traducció a llenguatge C i després a CUDA.

Uns altre tema que ha desviat la planificació del projecte, és la divergència dels resultats en els prototips desenvolupats, ja que la resposta esperada entre els prototips de Matlab i C donava uns valors parcials diferents. S'han fet proves unitàries en els entorns Matlab, C i CUDA que demostren que encara que implementin l'estàndard IEEE 745 (que defineix la representació en coma flotat), els resultats obtinguts en una operació unitària amb matrius dona una lleugera diferència entre les plataformes degut a cóm ressolen l'arrodoniment. Per tant, al llarg del procés i degut a les característiques de l'algorisme, els valors intermedis divergeixen lleugerament i provoquen un resultat inesperat, que és la convergència del proces entre plataformes diferents però amb un número d'iteracions no equivalent. Això vol dir que el procés de segmentació per un tall de la imatge pot convergir en 10 iteracions en una plataforma, i en una altre plataforma pel mateix tall pot fer-ho en 130, amb la qual cosa els algorisme no son vàlids per fer càlculs de temps, ja que no es comporten igual. Aquest comportament no desitjades veu afectat amb dades tipus double (64 bits) i més especialment en float (32 bits). Ja es sabia des de el començament que la tecnologia CUDA te una limitació amb les dades de tipus flotants, pel model hardware i versió software amb que s'ha desenvolupat el prototip, tot i així està previst per part d'NVIDIA, que les noves targetes gràfiques i les versions més actuals de software implementin el càlcul per dades tipus double. Per tant la limitació que ha aparegut al llarg del projecte es resoldria amb l'adquisició d'una nova targeta gràfica amb suport per tipus de dades flotant de doble precisió .

Altre singularitats de la tecnologia CUDA és que la part executada a la GPU (kernel) no suporta control de flux i que no es poden fer funcions recursives, i per tant s'han de sincronitzar les operacions i convertir les recursivitats en bucles, però això només serà una limitació pel programador que haurà de buscar altres estratègies d'implementació.

6.1 Planificació final

La planificació final reflexa desviacions i tasques no planificades a l'inici del projecte. Les noves etapes limiten l'abast del projecte i es centren en el procés de segmentació dins del tractament VBM (Voxel Based Morphometry) executat amb SPM (Statistical Parametric Mapping).

Fita 0 - Instal·lar l'entorn de treball. Manual d'instal·lació del software necessari. *Annex II*

Fita 1 - Estudi bibliogràfic general: Exposició en detall de l'estat de l'art i de les vies de futur.

Fita 2 – Preparació de la presentació inicial.

Fita 3 – Milestone presentació inicial. Presentació feta davant del departament de Neuroimatge.

Fita 4 – Formació entorn CUDA. Proves amb la plataforma CUDA

Fita 5 - Estudi del procés VBM. Desenvolupament d'un cas de VBM

Fita 6 – Anàlisi SPM. Estudi del software i disseny del processos que intervenen, es pot veure la documentació en *Annex I*

Fita 7 - Estudi Segmentació SPM. Anàlisis en profunditat del procés de segmentació SPM

Fita 8 - Implementació segmentació en C (basat en SPM). Substitució del codi per una funció MEX (C incrustat en Matlab) que és crida des de l'entorn Matlab.

Fita 9 - Anàlisis Segmentació Ashburner, Estudi de l'algorisme d'Ashburner per la segmentació.

Fita 10 - Implementació prototip Matlab. Implementació del codi en Matlab

Fita 11 - Implementació prototip C

Fita 12 - Implementació prototip CUDA versió 1, implementació paral·lela per talls

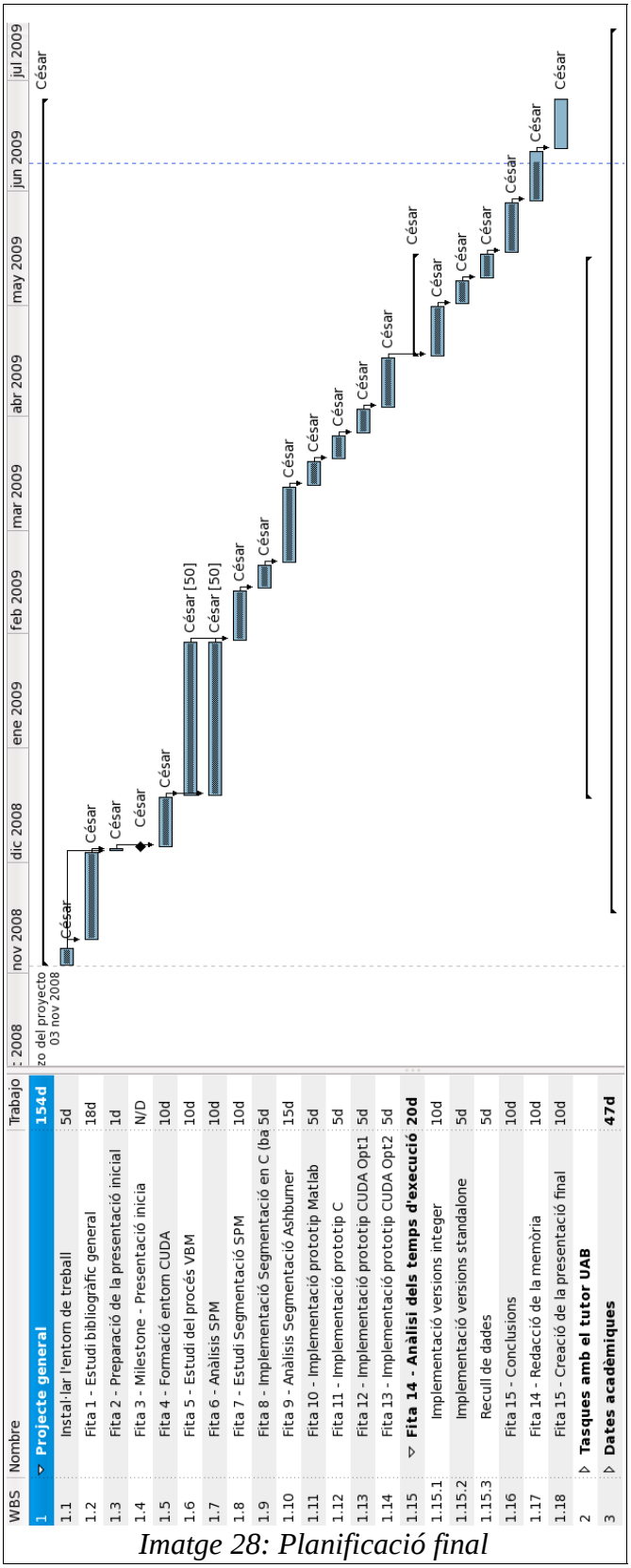
Fita 13 - Implementació prototip CUDA versió 2, implementació paral·lela per vòxels

Fita 14 - Anàlisi dels temps d'execució, execució en diverses plataformes hardware

Fita 15 - Conclusions

Fita 16 - Redacció de la memòria

Fita 17 - Creació de la presentació final



Imatge 28: Planificació final

6.2 Vies de futur

Durant el desenvolupament del projecte la tecnologia CUDA ha anat evolucionant constantment. Actualment el concepte de GPGPUs'està consolidant dins el món científic, i apareixen en el mercat nous productes software que aprofiten aquesta tecnologia i que pretenen simplificar el procés.

Algunes de les limitacions hardware que es van observar a l'inici del projecte, actualment estan superades, com per exemple la limitació de treball amb precisió simple, ja que està previst en noves generacions hardware i software, com per exemple, actualment a la redacció d'aquesta memòria ja es pot treballar amb dades en punt flotant de tipus double mitjançat les llibreries CUDA 2.1 i una entorn hardware amb servidors gràfics Tesla, de NVIDIA.

S'han estudiat altres tecnologies, com per exemple Jacket o GPUmat han estat estudiades, aquesta tecnologia és considera una passarel·la entre Matlab i CUDA, ja que permet desenvolupar codi Matlab executat en GPU. Tot i així en el moment de l'estudi ja estaba en una fase inicial del desenvolupament i es va considerar no viable.

El futur de la programació de propòsit general passa per OpenCL (Open Computing Language), la nova especificació que es vol convertir en estàndard. Desenvolupada pel Khronos Group (OpenGL, COLLADA,...) i recolzada per empreses com AMD, Apple,IBM, Intel, NVIDIA, Motorola, i altres. La aposta d'NVIDIA per OpenCL queda de manifest ja que ha estat la primera companyia en presentar una demo OpenCL executada en GPU. Una possible via es implementar el codi en OpenCL, i el cost no seria molt elevat ja que són llenguatges molt semblants, a diferencia d'altres tecnologies GPGPU existents, com AtiStream (Brook+), Close to Metal., etc. A més està previst que OpenCL estigui suportat independentment del hardware, i això permetria executar el codi en plataformes tant NVIDIA, com ATI, com altres.

7 Referències

[Anthony Gregerson] Anthony Gregerson, "Implementing Fast MRI Gridding on GPUs via CUDA"

[Antonio Ruiz et al.] Antonio Ruiz et al., "the GPU on biomedical image processing for color and phenotyphe analysis"

[Ashburner] John Ashburner, Image Segmentation, http://www.fil.ion.ucl.ac.uk/spm/doc/theses/john/chapter5.pdf, last visit on May 20, 2009

[CUDADoc] http://www.nvidia.es/object/cuda_education_es.html, last visit on 29 Gener de 2009

[CUDADownload] http://www.nvidia.es/object/cuda_get_es.html, last visit on 29 Gener de 2009

[CUDAProfiler] http://developer.download.nvidia.com/compute/cuda/2_1/cudaprof/CudaVisualProfiler_linux_1.1_15Dec08.tar.gz, *last visit on 10 de Juny de 2009*

[CUDASuport] http://www.nvidia.es/object/cuda_learn_products_es.html, last visit on 29 Gener de 2009

[David Field] David Field, *"Tesla news: CUDA vs OpenCL"*, http://www.atomicmpc.com.au/News/128547,tesla-news-cuda-vs-opencl.aspx , about a conference on November 7th 2008, last visit to the page on December 11th 2008

[Dominik Göddeke, Robert Strzodka, Stefan Turek] Dominik Göddeke, Robert Strzodka, Stefan Turek, *"Performance and accuracy of hardware -oriented native- emulated- and mixed-precision solvers in FEM simulations"*

[Friedemann Rößler et al.] Friedemann Rößler et al., *"GPU-based Multi-Volume Rendering for the Visualization of Functional Brain Images"*

[John D. Owens et al.] John D. Owens et al., *"A Survey of general-Purpose Computation on Graphics Hardware"*

[Mathworks] http://www.mathworks.com/, last visit on 29 Gener de 2009

[Michael Wolfe] Michael Wolfe, *"How we should program GPGPUs"*, linuxjournal , november 2008

[MRI_TOOLBOX] Jimmy Shen, http://www.mathworks.com/matlabcentral/fileexchange/8797, last visit on 5 de Juny de 2009

[Olaf Schenk, Matthias Christen, Helmar Burkhart] Olaf Schenk, Matthias Christen, Helmar Burkhart, *"Algorithmic Performance Studies On Graphics Processing Units"*

[OpenCL demo] First OpenCL demo on a GPU, http://www.youtube.com/watch?v=r1sN1ELJfNo&feature=channel_page, l*ast visit on 20 Maig de 2009*

[Rob Farber] Rob Farber, "Feature CUDA", linuxjournal, november 2008

[Sam S. Stone et al.] Sam S. Stone et al., "How GPUs Can Improve the Quality of Magnetic

Rosonance imaging"

[Shane Ryoo et al.] Shane Ryoo et al., *"Program Optimization Space Pruning for a Multithreaded GPU"*, Center for Reliable and High-Performance Computing, University of Illinois at Urbana-Champaign

[SPM] http://www.fil.ion.ucl.ac.uk/spm/, last visit on 29 Gener de 2009,

[T1 templates] Alan Evans, John Mazziotta, ,http://imaging.mrc-cbu.cam.ac.uk/imaging/Templates T1 templates, last visit on May 20,2009

[Timothy D. R. Hartley et al.] Timothy D. R. Hartley et al., *"Biomedical Image Analysis on a Cooperative Cluster of GPUs and Multicores"*

[Webinar GPU] "GPU Computing Tutorials", web seminar on April 15, 2009 at 11am PST

[Webminar Jacket] Sumit Gupta, Ph. D, Sr Product Manager of Tesla GPU Computing at NVIDIA and John Melonakos, Ph.D CEO at AccelerEyes LLC *"Jacket: Accelerating MATLAB using CUDA-enabled GPUs",* web seminar on February 15, 2009

[Webinar Performance CUDA] Nadeem Mohammad, *"Performance considerations for CUDA programming"*, web seminar on May 6, 2009 at 11:00am

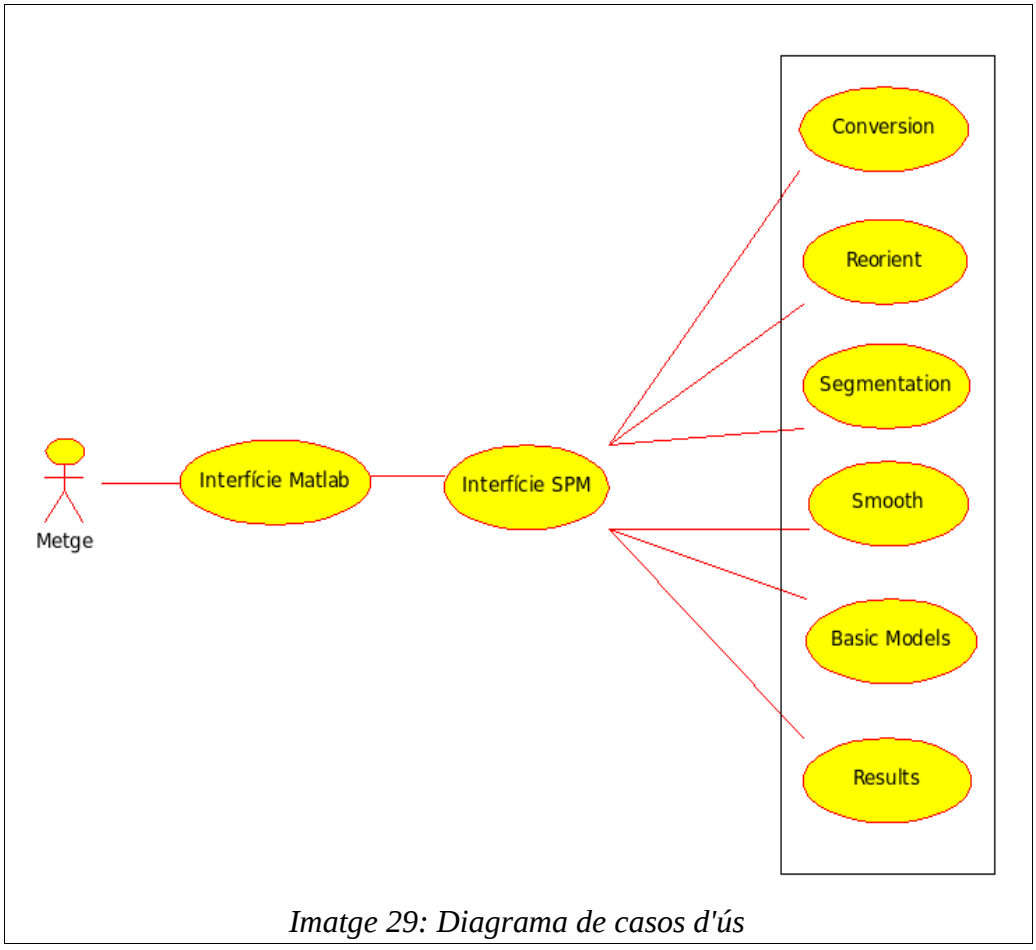
Estudi GPGPU dedicat al processament de neuroimatges

ANNEXE I - Anàlisis del procés VBM amb SPM

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per César Allande Álvarez
i dirigit per Diego Mostaccio

1 Descripció Casos d'Ús

A continuació es defineixen les especificacions detallades de cada cas d'ús representades en en el diagrama de cosos d'ús de la *imatge 29.*



Imatge 29: Diagrama de casos d'ús

Cas d'ús Interfície Matlab

Descripció:

Executarà la crida per iniciar el paquet SPM. Es realitzarà mitjançant la crida a la funció Matlab mitjançant la comanda *spm*. Aquesta acció ens obre una finestra de benvinguda on es pot escollir l'entorn de treball:

- PET (Positron Emission Tomography) and SPECT. Comanda: *spm('PET')*
- EEG (Electroencephalography). Comanda: *spm('EEG')*
- fMRI (Functional Magnetic Resonance Imaging).Comanda: *spm('fMRI')*

També es permet passar com a paràmetre el nom de l'entorn per accedir-hi directament.

Actors:

L'actor amb rol de metge.

Precondicions

Tenir llicència de Matlab corresponent, i tenir executant el software de Matlab.

Conèixer comandes bàsiques de Matlab.

Flux Normal

Executar la comanda que inicia SPM des de la línia de comandes.

Si es passa un paràmetre per entrar directament en un entorn gràfic aquest és el paràmetre 'PET'.

Poscondicions

Si s'escriu la comanda s'obrirà la interfície SPM.

Cas d'ús Interfície SPM

Descripció:

Si s'ha iniciat amb la comanda sense paràmetre, inicialment es mostrarà una pantalla prèvia de benvinguda que permet a l'actor seleccionar l'entorn amb el que desitja treballar. Els entorns que es poden escollir són PET, fmRi, EEG. Cadascun dels entorns té una configuració gràfica diferent.

Un cop seleccionat l'entorn de treball la configuració de les finestres canvia i es mostren les tres finestres de l'entorn seleccionat:

- Finestra superior esquerra, conté els botons que executen les funcions de posprocessament d'imatges.
- Finestra inferior esquerra, mostra l'evolució de l'execució en temps real del procés que s'està executant.
- Finestra dreta, pantalla de configuració del processos. Es poden veure, modificar i carregar els paràmetres dels jobs que es llençaran.

Actors:

L'actor amb rol de metge selecciona el procés que vol realitzar seleccionant el botó en la finestra superior esquerra. La configuració de la finestra dreta canviarà per mostrar els paràmetres del procés.

L'actor ha de tenir coneixements de posprocessament d'imatges mèdiques, per configurar els valors dels paràmetres.

Precondicions

Tenir activada la interfície Matlab, i l'entorn visual SPM.

Flux Normal

L'actor ha de polsar els botons seleccionant el procés que vol desenvolupar, i a la finestra de selecció de paràmetres ajustar els valors del seu job. L'usuari pot carregar, o guardar configuracions. Finalment es polsa el botó 'Run' per llençar el procés.

Poscondicions

Si s'ha polsat el botó 'Run', el procés s'executarà i es podrà veure l'evolució del procés a la pantalla inferior esquerra.

Cas d'ús Conversion

Descripció:

Seleccionar els fitxer en format DICOM que es volen convertir a format NIfTI.

Actors

L'actor amb rol de metge

Precondicions

Estar a dins de la interfície SPM.

Disposar d'imatges en format DICOM.

Flux Normal

Polsar el botó 'DICOM Imp...'. Seleccionar els arxius DICOM. Seguidament apareixerà una pantalla per la selecció de l'arxiu de destí. Seguidament es mostra a la pantalla inferior esquerra un menú de selecció que permet escollir el format de sortida (NIfTI o Analyze). Es selecciona 'NIfTI'. El menú de selecció canvia i pregunta 'Use ICEDims in filename...', i es selecciona 'No'

Flux Alternatiu

Es podrà seleccionar el format de sortida Analyze.

Cas d'ús Reorient

Descripció:

Ajustar manualment les imatges, una a una, per intentar alinear-les el màxim possible i centrar-les en el mateix punt de l'espai tridimensional.

Actors:

L'actor amb rol de metge

Precondicions

Estar a dins de la interfície SPM.

Tenir imatges en format NIfTI o Analyze.

Flux Normal

Polsar el botó 'Display'. Seleccionar la imatge a reorientar i un cop visualitzada es poden modificar els paràmetres. Els valors de right,forward i up, permeten desplaçar la imatge en mil·límetres. Els valors de pitch, roll i yaw, fan rotar la imatge. Els valors de resize(x),resize(y) i resize(z) per escalar la imatge. Un cop ajustada es polsa el botó 'Check Reg', que guarda la imatge reorientada.

Flux Alternatiu

En qualsevol moment es pot tornar a la orientació original polsant el botó 'Origin'.

Cas d'ús Segmentation

Descripció:

Procés automàtic que permet segmentar la/les imatge/s i generar-ne de noves amb informació extreta de la original. La informació que es vol extreure pot ser matèria gris, matèria blanca o líquid cefaloraquidi.

Actors:

L'actor amb rol de metge.

Precondicions

Estar a dins de la interfície SPM.

Disposar d'imatges en format NIfTI o Analyze. Es aconsellable fer una reorientació del conjunt d'imatges a segmentar.

Flux Normal

Polsar el botó 'Segment'. Seleccionar la/les imatge/s a segmentar i ajustar els paràmetres. Es pot especificar en els paràmetres el resultat que es vol obtenir activant les caselles de Grey Matter, White Matter, Cerebro-Spinal fluid. Polsar el botó 'Run' per iniciar el procés.

Flux Alternatiu

Modificació de la resta de paràmetres.

Guardar, carregar i/o executar altres configuracions.

Poscondicions

Es genera un arxiu nou per cada casella activada de Grey Matter, White Matter, Cerebro-Spinal fluid.

Els fitxer generats són:

- Grey Matter: Modulated (mwc1*.img), unmodulated (wc1*.img)
- White Matter: Modulated (mwc2*.img), unmodulated (wc2*.img)
- Cerebro-Spinal fluid: Modulated (mwc3*.img), unmodulated (wc3*.img)

Cas d'ús Smooth

Descripció:

Procés automàtic que ajustant els paràmetres permet aplicar un filtre a la/les imatge/s. Aquest procés permet filtrar el promig d'intensitat dels vòxels adjacents.

Actors:

L'actor amb rol de metge.

Precondicions

Estar a dins de la interfície SPM.

Flux Normal

Polsar el botó 'Smooth'. Seleccionar la/les imatge/es a filtrar. Especificar el paràmetre FWMM (Full Width at Half Maximum). Polsar el botó 'Run' per iniciar el procés.

Flux Alternatiu

Modificació de la resta de paràmetres.

Guardar, carregar i/o executar altres configuracions.

Poscondicions

Es genera una imatges de sortida per cada imatge d'entrada. El nom del fitxer de la nova imatge comença per 's'.

Cas d'ús Basic Models

Descripció:

Procés automàtic que ajustant els paràmetres permet generar un model estadístic que permet expressar la variable depenent de l'estudi.

Actors:

L'actor amb rol de metge.

Precondicions

Estar a dins de la interfície SPM. Disposar d'imatges en format NIfTI o Analyze.

Flux Normal

Polsar el botó 'Basic Models'. Seleccionar les imatges dels dos grups d'estudi. Especificar el

paràmetres estadístics. Polsar el botó 'Run' per iniciar el procés.

Flux Alternatiu

Modificació de la resta de paràmetres. Guardar, carregar i/o executar altres configuracions.

Poscondicions

S'ha de generar un arxiu {SPM.mat amb els resultats. S'ha de mostrar un gràfic amb la representació de l'estudi realitzat.

Cas d'ús Results

Descripció:

Procés automàtic pel tractament de la estadística del procés Basic Models. Mostrarà els grups, covariables i tipus d'estadística en la finestra de la dreta. Les dades resultants presenten les zones de creixement/deteriorament de la matèria.

Actors:

L'actor amb rol de metge.

Precondicions

Estar a dins de la interfície SPM.

Disposar del model estadístic SPM.mat generat en el procés Basic Models.

Flux Normal

Polsar el botó 'Results'. Seleccionar l'arxiu SPM.mat generat en Basic Models. S'iniciarà el procés.

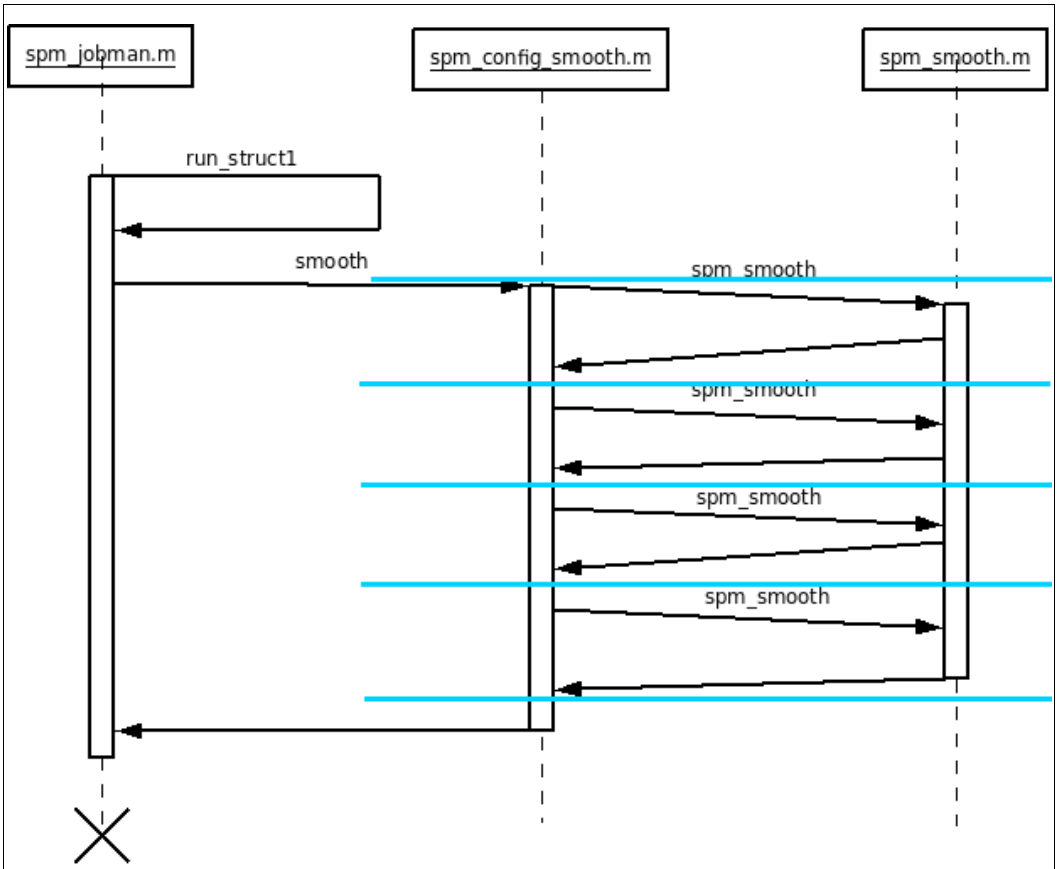
Poscondicions

Es mostrarà el model generat en la pantalla de la dreta.

2 Procés Smooth

El procés realitza un suavitzat de la imatge.

Diagrama de Seqüència En aquest diagrama de seqüència es delimiten mitjançant una línia blava el procés Smooth per les 4 imatges.



Imatge 30: Diagrama de Seqüència de Smooth. x4 imatges

Temps d'execució Aquí tenim els temps del procés Smooth generat per 4 imatges, on en principi s'hauria de mantenir la relació per màquina del mateix ordre x5.

Arxiu	**Funció**	**Temps DELL XPS**		**Temps Thosiba Lap**	
		segons	**minuts**	**segons**	**minuts**
spm_jobman.m	*run_struct*	1,13	0,02	130,05	2,17
spm_jobman.m	*run_struct1*	1,13	0	130,05	2,17
spm_config_smooth.m	*smooth*	1,11	0	130,02	2,17
spm_smooth.m	*spm_smooth #1 image*	0,23	0	23,36	0,39
spm_smooth.m	*spm_smooth #2 image*	0,23	0	23,56	0,39
spm_smooth.m	*spm_smooth #3 image*	0,23	0	20,78	0,35
spm_smooth.m	*spm_smooth #4 image*	0,23	0	62,17	1,04

Taula 18: Temps d'execució del procés Smooth, en SPM

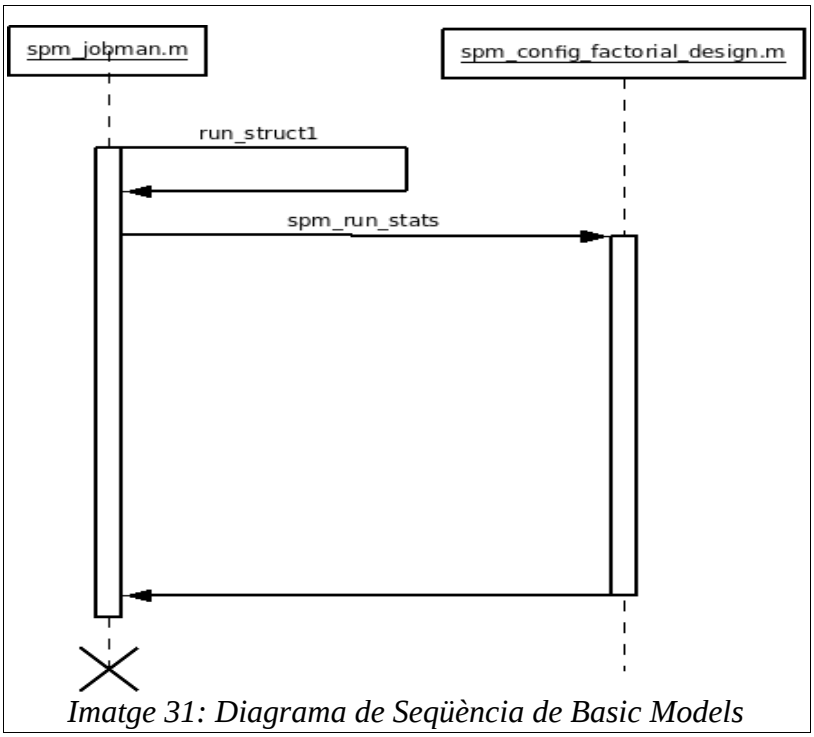
Els temps executats en la màquina DELL XPS mostren que aquesta funció no triga molt temps en ser executada, a diferència del temps amb el Laptop Thosiba, que triga unes 130 vegades més. No es manté la relació de x5 del procés Segmentation.

En veure que els temps no són molt grans, es descartarà en principi el procés Smooth, ja que es considera que no tindrem un guany substancial al fer un codi paral·lel. En cas de necessitar millorar els temps la funció crítica a modificar serà spm_smooth del fitxer spm_smooth.m.

3 Procés Basic Models

Aquest procés genera el model estadístic basat en les imatges dels grups de control i d'estudi.

Diagrama de Seqüència, aquest procés accedeix a dues funcions.



Imatge 31: Diagrama de Seqüència de Basic Models

Temps d'execució En l'execució del procés Basic Models, tota la càrrega de càlcul recau en el procés spm_confing_factorial_design.

Arxiu	Funció	Temps DELL XPS		Temps Thosiba Lap	
		segons	minuts	segons	minuts
spm_jobman.m	*run_struct*	1,18	0,02	0,72	0,01
spm_jobman.m	*run_struct1*	1,18	0,02	0,72	0,01
spm_config_factorial_design.m	*execute*	1,14	0,02	6,67	0,11

Taula 19: Temps d'execució del procés Basic Models, en SPM

Aquest procés no consumeix molt temps. En cas de necessitar millorar el temps la funció a crítica a modificar serà execute de spm_config_factorial_design.

Estudi GPGPU dedicat al processament de neuroimatges

ANNEXE II - Manual instal·lació CUDA

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per César Allande Álvarez
i dirigit per Diego Mostaccio

1 Manual instal·lació CUDA

En aquest document es recull la informació que s'ha anat recopilant durant la instal·lació de l'entorn CUDA per la realització del projecte “1069 - Desenvolupament d’una plataforma GPGPU dedicada al processament de neuroimatges". El software necessari pel desenvolupament del projecte ha estat:

- CUDA - Compute Unified Device Architecture
- Matlab - MATrix LABoratory
- SPM5 – Statistical Parametric Mapping v.5
- CUDA plugin for Matlab

2 CUDA

CUDA proporciona les llibreries de programació GPGPU, que permeten executar codi de propòsit general en la GPU de la tergeta gràfica. A la pàgina *[CUDADoc]* podem trovar tota la documentació referent a CUDA.

Aquesta tecnologia pertany a empresa NVIDIA, actualment líder en el mercat de les targetes gràfiques. Fins a la data la flexibilitat del desenvolupament d'aplicacions amb CUDA està limitat a una gama de targes gràfiques d'NVIDIA i que podem trobar a la pàgina oficial d'NVIDIA *[CUDASuport].*

Si acomplim els requeriments hardware podem seguir amb la instal·lació del software adequat al nostre entorn de treball. En el nostre cas la instal·lació s'ha fet en un entorn Linux, amb la distribució Fedora 10 de 32 bits. Els programes es poden descarregar de [*CUDADownload].* Per realitzar una instal·lació correcta seguirem l'ordre indicat a les especificacions:

- Driver NVIDIA amb suport CUDA, per Fedora

- Toolkit, llibreries CUDA
- CUDA SDK, Entorn de desenvolupament amb exemples CUDA
- CUDA Visual Profiler, permet mesurar els temps d'execució

A la mateixa pàgina també es podem trobar un Debugger i el seu manual d'utilització.

3 Driver NVIDIA amb suport CUDA

Per instal·lar el driver en Linux Fedora, necessitem executar l'instal·lador del driver des de una terminal com a superusuari. Obrim una terminal(Ctrl+Alt+F1):

 # su -

Ens identifiquem com a root.

 # init 3

Mode complet multiusuari, sense entorn gràfic.

 # chmod +x NVIDIA-Linux-x86-180.06-pkg1.run

Permís d'execució

 # ./NVIDIA-Linux-x86-180.06-pkg1.run

Iniciar l'nstal·lació del driver

Es segueixen els passos que indica l'instal·lador, i s'inicia la compilació del mòdul. Un cop acabat, es pot iniciar el mode 5 per verificar que el driver funciona correctament.

 # init 5

NOTA:

Abans d'inicial l'execució podem resoldre les dependències per la compilació del modul. Farem

servir el gestor de paquets yum per instal·lar-les mitjançant aquesta comanda.

 # yum instal gcc-c++ freeglut-devel libXi-devel libXmu-devel

4 Toolkit, llibreries CUDA

Per instal·lar el toolkit també hem de donar-li permisos d'execució:

 # chmod +x cuda-linux-rel-nightly-2.1.1635-3065709.run

 # ./cuda-linux-rel-nightly-2.1.1635-3065709.run

Bàsicament copia a la carpeta de destí l'estructura jeràrquica de codi font i llibreries. L'adreça de destí per defecte és:

/usr/local/cuda.

Al finalitzar el procés s'ens demana afegir les adreces de les llibreries CUDA a les variables d'entorn. Per fer-ho afegirem al ~/.bashrc del home del desenvolupador les següents línies:

export PATH=$PATH:/usr/local/cuda/bin

export LD_LIBRARY_PATH=/usr/local/cuda/lib

5 CUDA SDK

El SDK de CUDA conté codi d'exemple. Per instal·lar el CUDA SDK li donarem permisos d'execució.

 # chmod +x cuda-sdk-linux-2.10.1126.1520-3141441.run

 # ./cuda-sdk-linux-2.10.1126.1520-3141441.run

El directori per defecte on s'instal·la el SDK és

~/NVIDIA_CUDA_SDK

Per verificar que la instal·lació de CUDA és correcte realitzarem la compilació dels exemples. Per fer-ho només cal entrar a la carpeta del SDK i compilar mitjançant la configuració del Makefile:

 # cd ~/NVIDIA_CUDA_SDK

 # make

NOTA: Si falla la compilació, i ens mostra un error de la llibreria glut, pot ser que manqui una referència que es pot solucionar executant :

sudo ln -s /usr/lib/libglut.so.3 /usr/lib/libglut.so

Aquesta dependència s'explica al document de les notes de distribucció *[SDKReleaseNotes]*

Un cop finalitzada correctament la compilació podem executar els exemples que es troven a la carpeta:

~/NVIDIA_CUDA_SDK/bin/linux/release/

per exemple:

 # ~/NVIDIA_CUDA_SDK/bin/linux/release/oceanFFT

6 CUDA Visual Profiler

El Visual Profiler ens permet fer descarregar, primerament el descomprimim i el movem al directori

corresponent:

 # tar -xvzf CudaVisualProfiler_linux_1.1Beta_13Nov08.tar.gz

 # mv CudaVisualProfiler /usr/local/CudaVisualProfiler

NOTA :

Dependències de l'entorn gràfic amb les llibrerires Qt.

 # yum install qt-devel

Afegim CudaVisualProfiler a les variables d'entorn LD_LIBRARY_PATH

export PATH=$PATH:/usr/local/cuda/bin

export LD_LIBRARY_PATH=/usr/local/cuda/lib:/usr/local/CudaVisualProfiler/bin

Creem un enllaç a ./cudaprof

sudo ln -s /usr/local/CudaVisualProfiler/bin/cudaprof /usr/bin/cudaprof

7 Matlab

Per la instal·lació de Matlab 7r2008a, s'ha seguit el procés explicat a la documentació adjunta al software original.

Hem obtingut la llicència i el número d'activació de la pàgina web de *[Mathworks]*. Per fer-ho hem entrat a la pàgina web amb el nostre usuari i password i a la secció de llicències hem activat una de les llicències disponible. Això ens permet descarregar-nos el *product key license* (clau per

instal·lar el software), i la *license.dat*.

NOTA:

L'instal·lador de Matlab necessita la dependència:

yum install libXp-devel

Inserim el CD i executem l'arxiu install. Seguim els passos que ens demana l'instal·lador. La nostra ruta per defecte de Matlab és:

/usr/local/matlab

Comprovem que la instal·lació s'ha executat correctament, teclejant la comanda per llençar Matlab

 # matlab

8 SPM

Per instal·lar SPM5 *[SPMDownload]* ens descarreguem el software de la pàgina oficial *[SPM]*, després d'omplir el formulari.

Descomprimim i copiem a la ruta.

 # tar -xvzf SPM.tar.gz

*# mv SPM /usr/local/SP*M

Després hem d'afegir la referència al PATH de Matlab del paquet SPM, per tant executem Matlab com a superusuari.

 # su -

 # matlab

File --> Set Path --> botó "Add with Subforlders"

Seleccionem la ruta */usr/local/SPM*

Botó "Save"

A l'interpret de comandes de Matlab ja podem executar SPM.

>> spm

9 CUDA Plugin for Matlab

Descarreguem el Plugin de la pàgina de CUDAZone (Fòrum de CUDA), *[CUDAplugin].*

 # tar -xvzf Matlab_Cuda_1.1.tgz

 # mv Matlab_Cuda /usr/local/MatlabCuda

Repetim els passos per afegir una adreça al PATH de Matlab

 # su -

 # matlab

File --> Set Path --> botó "Add with Subfolders"

Seleccionem la ruta /usr/local/MatlabCuda

Botó "Save"

Ara podem compilar un exemple de Matlab amb CUDA, ubicat a la carpeta de MatlabCuda, anomenat *README.TXT*.

>> which Szeta

>> tic; FS_2Dturb(128,1,1,1); toc;

Veiem el temps que ha trigat pel codi matlab

>> unix('make');

>> which Szeta

>> tic; FS_2Dturb(128,1,1,1); toc;

Podem comparar el temps que ha trigat el .mexglx respecte a la versió anterior .m-

10 Jacket

El producte Jacket d'Accelereyes permet programar amb el llenguatge M de Matlab efectuant operacions a GPU. Aquesta llibreria s'ha provat i ha donat bons resultats.

Per realitzar la instal·lació´o s'ha instal·lat una versi´o de 15 dies de prova. I S'ha seguit el tutorial d'instal·lació´o de la llibreria.

11 Notes post-instal·lació

Si tenim el gestor d'actualitzacions de Fedora s'ha de tenir en compte que cada vegada que s'actualitzi el kernel, la targeta gràfica ens pot donar problemes de configuració.

La solució que s'ha seguit en el desenvolupament del projecte és desactivar les actualitzacions automàtiques.

També es pot reconfigurar els drivers amb l'instal·lador, ja que aquest torna a compilar els drivers pel nou kernel. S'ha de tenir en compte que cada vegada que s'actualitzi el kernel haurem de torar a instalar l'últim driver disponible a la pàgina web d'NVIDIA.

12 Referències del manual

[CUDADoc] http://www.nvidia.es/object/cuda_education_es.html, last visit on 29 Gener de 2009

[CUDADownload] http://www.nvidia.es/object/cuda_get_es.html, last visit on 29 Gener de 2009

[CUDAplugin] http://developer.nvidia.com/object/matlab_cuda.html, last visit on 29 Gener de 2009

[CUDASuport] http://www.nvidia.es/object/cuda_learn_products_es.html, last visit on 29 Gener de 2009

[Mathworks] http://www.mathworks.com/, last visit on 29 Gener de 2009, last visit on 29 Gener de 2009

[SDKReleaseNotes]
http://moss.csc.ncsu.edu/~mueller/cluster/nvidia/2.0/SDK_Rel_Notes_Linux_2.0beta2.txt, last visit on 29 Gener de 2009

[SPM] http://www.fil.ion.ucl.ac.uk/spm/, last visit on 29 Gener de 2009, last visit on 29 Gener de 2009

[SPMDownload] http://www.fil.ion.ucl.ac.uk/spm/software/download.html, last visit on 29 Gener de 2009

Estudi GPGPU dedicat al processament de neuroimatges

ANNEXE III - Glossari

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per César Allande Álvarez
i dirigit per Diego Mostaccio

ATI-Stream – Entorn GPGPU elaborat per la companyia ATI-AMD. Esta integrat en un llenguatge d'alt nivell, a diferència de CTM.

Comissura anterior, Porció de la matèria blanca del cervell. Situada davant dels Pilars anteriors del Trígon i darrera de la làmina Terminal Cinerea o del Pic del cos callós. Es fa servir com a punt de referència a l'hora d'ubicar dues imatges en el mateix punt estereotàxic

CTM (Close To Metal) – Entorn GPGPU elaborat per la companyia ATI-AMD.

CUDA (Compute Unified Device Architecture) – Entorn GPGPU elaborat per la companyia NVIDIA. Llenguatge d'alt nivell integrat en C.

Estereotàxic, espai – Per situar dues imatges de RM (ressonància magnètica) en el mateix espai estereotàxic, s'agafa un punt de referència on fixar les coordenades del punt cero, normalment es fa servir la comissura anterior.

GPGPU (General Purpose on GPU) – Propòsit general en xips gràfics. Permet el coprocessament de dades a través de la targeta gràfica.

Matlab (Matrix Laboratory) – Software matemàtic desenvolupat per Mathworks.

SPM (Statistical Parametric Mapping) – Paquet Matlab dedicat al tractament de neuroimatges. Desenvolupat per membres i col·laboradors del Wellcome Trust Centre for Neuroimaging de Londres.

Vòxel – Píxel volumètric. Cada element de la imatge 3D.

Voxel Based Morphometry – Tractament per detectar diferències estructurals en el cervell, es basa en un model probabilistic.

Signat:

Cèsar Allande Álvarez

Resum:

Estudi GPGPU dedicat al processament de neuroimatges. La tecnologia GPGPU permet paral·lelitzar càlculs executant operacions aritmètiques en els múltiples processadors de que disposen els xips gràfics. S'ha fet servir l'entorn de desenvolupament CUDA de la companyia NVIDIA, que actualment és la solució GPGPU més avançada del mercat. L'algorisme de neuroimatge implementat pertany a un estudi VBM desenvolupat amb l'eina SPM. Es tracta concretament del procés de segmentació d'imatges de ressonància magnètica cerebrals, en els diferents teixits dels quals es composa el cervell: matèria blanca, matèria grisa i líquid cefaloraquidi. S'han implementat models en els llenguatges Matlab, C i CUDA, i s'ha fet un estudi comparatiu per plataformes hardware diferents.

Resumen:

Estudio GPGPU dedicado al procesamiento de neuroimágenes. La tecnología GPGPU permite paralelizar cálculos ejecutando operaciones aritméticas en los múltiples procesadores de que disponen los chips gráficos. Se ha utilizado el entorno de desarrollo CUDA de la compañía NVIDIA, que actualmente es la solución GPGPU más avanzada del mercado. El algoritmo de neuroimagen implementado pertenece a un estudio VBM desarrollado con la herramienta SPM. Se trata específicamente del proceso de segmentación de imágenes de resonancia magnética cerebrales, en los diferentes tejidos de los que se compone el cerebro: materia blanca, materia gris y líquido cefalorraquídeo. Se han implementado modelos en los lenguajes Matlab, C y CUDA, y se ha hecho un estudio comparativo para plataformas hardware diferentes.

SUMMARY:

Study GPGPU focused on neuroimage processing. GPGPU technology allows parallelization of calculations executing arithmetic operations in any of the multiple processors that graphic chips have. The environment used for development is CUDA from NVIDIA, which is currently the most advanced solution available. The neuroimage algorithm implemented is part of a VBM study developed with the SPM tool. Therefore, the goal of this project is the process of segmentation of neural magnetic resonances into the different tissues the brain is composed of: white matter, grey matter and cefalorraquid liquid. Different models have been implemented in Matlab, C and CUDA, and finally, a comparative study between different hardware platforms.